NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

AN ANALYSIS OF THE TOKEN RING PROTOCOL
AS SPECIFIED IN ANSI/IEEE STANDARD 802.5-1985

by

Nejdet AYIK

March 1989

Thesis Advisor G.M. Lundy

Unclassified

security classification of this page

REPORT DOCUMENTATION PAGE

1a Report Security Classification: Unclassified			1b Restrictive Markings
2a Security Classification Authority			3 Distribution/Availability of Report: Approved for public release: distribution is unlimited.
2b Declassification/Downgrading Schedule			
4 Performing Organization Report Number(s)			5 Monitoring Organization Report Number(s)
6a Name of Performing Organization: Naval Postgraduate School	6b Office Symbol (if applicable): 32		7a Name of Monitoring Organization: Naval Postgraduate School
6c Address (city, state, and ZIP code): Monterey, CA 93943-5000			7b Address (city, state, and ZIP code): Monterey, CA 93943-5000
8a Name of Funding/Sponsoring Organization	8b Office Symbol (if applicable)		9 Procurement Instrument Identification Number
8c Address (city, state, and ZIP code)			10 Source of Funding Numbers: Program Element No / Project No / Task No / Work Unit Accession No

11 Title (include security classification): AN ANALYSIS OF THE TOKEN RING PROTOCOL AS SPECIFIED IN ANSI/IEEE STANDARD 802.5-1985

12 Personal Author(s): Nejdet AYIK

13a Type of Report	13b Time Covered	14 Date of Report (year, month, day)	15 Page Count
Master's Thesis	From To	March 1989	74

16 Supplementary Notation: The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 Cosati Codes			18 Subject Terms (continue on reverse if necessary and identify by block number)
Field	Group	Subgroup	Token ring, communication protocols, formal modeling techniques, computer networks, local area networks.

19 Abstract (continue on reverse if necessary and identify by block number)

This thesis discusses the formal specification techniques for communication protocols and the ANSI/IEEE Standard 802.5 "Token Ring Access Method and Physical Layer Specifications." Background information on formal protocol specification and a review of the targeted standard are provided. The ambiguities that were found with the standard and solutions to some of those are presented. The study concludes that there is a growing need to find methods which will provide correct, clear and unambiguous methods for the specification and analysis of communication protocols and standards.

20 Distribution/Availability of Abstract	21 Abstract Security Classification	
☒ unclassified/unlimited ☐ same as report ☐ DTIC users	Unclassified	
22a Name of Responsible Individual: G.M. Lundy	22b Telephone (include Area code): (408) 646-2094	22c Office Symbol: 52Ln

DD FORM 1473,84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete

security classification of this page

Unclassified

Approved for public release; distribution is unlimited.

An Analysis of the Token Ring Protocol
as Specified in ANSI IEEE Standard 802.5-1985

by

Nejdet AYIK
First Lieutenant, Turkish Army
BS, Army Academy, 1982

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN TELECOMMUNICATIONS SYSTEMS MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
March 1989

Author: _____
Nejdet AYIK

Approved by: _____
G.M. Lundy, Thesis Advisor

_____
William Gates, Second Reader

_____
David R. Whipple, Chairman,
Department of Administrative Sciences

_____
Kneale T. Marshall,
Dean of Information and Policy Sciences

ABSTRACT

This thesis discusses the formal specification techniques for communication protocols and the ANSI/IEEE Standard 802.5 "Token Ring Access Method and Physical Layer Specifications." Background information on formal protocol specification and a review of the targeted standard are provided. The ambiguities that were found with the standard and solutions to some of those are presented. The study concludes that there is a growing need to find methods which will provide correct, clear and unambiguous methods for the specification and analysis of communication protocols and standards.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

The need for *data communications* existed even before the invention of computers. The concept of computer networks has evolved from the growing demand for remote computing features, some of which are:

- data exchange between systems,
- sharing expensive resources,
- ability to access devices remotely,
- backup facilities for real-time applications. [Ref. 1]

Computer networks can be classified into different categories based on how the data are processed and transmitted through the network (*data transfer technique*), how far the devices are physically separated (*geographical coverage*), or how the communicating devices are connected (*topology*). [Ref. 2]

Physical separation of the devices defines three types of networks:

- Wide Area Network (WAN):

  Devices are distributed over a wide geographical area, physical separation is usually more than 10km (i.e., countrywide or worldwide) [Ref. 3: p.6].

- Metropolitan Area Network (MAN):

  Devices are distributed over a metropolitan area, within a diameter of up to 50km (i.e., citywide) [Ref. 4: p.2].

- Local Area Network (LAN):

  Devices are distributed over a localized area, physical separation is less than 1km (i.e., within a single building or a group of localized buildings) [Ref. 3: p.6].

Among the above categories LANs play a key role, for in many cases they are the nodes of a WAN. "LANs are particularly important in that it is a LAN that will be connected to many workstations as the first stage in a larger distributed networking and computing environment." [Ref. 4 : p.1]

There are three common topologies used for LANs : *Bus/tree, star,* and *ring*. Each has strengths and weaknesses. Some of the benefits ring topology has over others are:

- minimized transmission errors,
- longer distance coverage,
- ability to accommodate high speed optical fiber links,
- simpler electronics and maintenance,
- automatic acknowledgement ability,
- better throughput with least sensitivity to work load. [Ref. 5: pp.349-367]

Although there have been a number of *medium access control* (MAC) techniques proposed for ring topology, the Token Ring access method has become the most popular one and it is the one which has been selected for standardization by IEEE 802 Local Network Standards Committee [Ref. 5: p.355].

A. PURPOSE OF THE STUDY

In order for data communications to take place, a set of requirements has to be fulfilled. Along with the apparent need for the intelligent devices, referred to as *data terminal equipment* (DTE), there must exist some sort of media through which the signals can propagate to and from the devices. Transmission through the media is subject to errors, and they should be detected and corrective actions taken. The messages must be broken down into appropriate units that can be processed and transmitted as signals, and the received signals have to be reconstructed into the original form of the messages. All of these requirements lead to a key requirement; there must be a common "language" which the DTEs use for communication. This common language, a set of complex rules or algorithms which have to be followed for a successful communication, is known as a *communication protocol* [Ref.6: p.2]. The essence of protocols is to ensure that pieces of the system work as a harmonious whole [Ref. 7: p.46].

Considering the complexity of networks and the variety of components produced by different vendors, the importance of correct, clear and unambiguous protocols is obvious. Protocol specification and analysis techniques have been a major research subject and are likely to stay that way for some time.[Ref. 7: p.51]

However, improvements in modeling and specification techniques alone are not sufficient to overcome problems with the heterogeneity of communications products. There is also a need for a mechanism to ensure that equipment from different vendors will communicate without a requirement for major protocol conversion means. This mechanism is the "standard." Although standards tend to slow down technological advance, the advantages they provide have led the communications and computer communities to welcome them. Today almost all areas of communications technology are governed by standards. [Ref. 5: pp.12-14]

An understanding of the importance of protocols in computer networks has urged the author to study the formal protocol specification techniques. The purpose of this thesis is to work on a recently written protocol standard, the "ANSI/IEEE Standard 802.5, Token Ring Access Method and Physical Layer Specifications." and determine whether there are any ambiguities or problems, and if so whether they can be clarified and solved.

B. ORGANIZATION OF THE THESIS

This chapter serves as an introduction.

Chapter II, Background: provides an overview of protocol specification and analysis techniques including an emphasis on Communicating Finite State Machines and Extended Finite State Machines. A general functional description of Token Ring is also given.

Chapter III, A Review of the IEEE Standard 802.5 : discusses the ANSI/IEEE Standard 802.5-1985 Token Ring Access Method and Physical Layer Specifications.

Chapter IV, Problems and Ambiguities: points out the problems and ambiguities which were found in the standard.

Chapter V, Suggested Solutions: provides suggested solutions for the identified problems.

Chapter VI, Conclusions and Recommendations: summarizes the conclusions reached by this study along with suggestions for further work.

II. AN OVERVIEW OF PROTOCOL MODELING AND THE TOKEN RING NETWORK

A. METHODS CURRENTLY USED IN PROTOCOL MODELING

There have been numerous studies on formal modeling of protocols. Rudin [Ref. 7] provides a discussion on the importance of formal modeling.

The methods for modeling protocols can be categorized into one of the following:

- Communicating Finite State Machines (CFSM),
- Extended Finite State Machines (EFSM)
- Programming Languages,

The following sections review the commonly used methods with an emphasis on the CFSM method, as it appears to be the basis for other techniques.

1. Communicating Finite State Machines

The CFSM models have the advantage of ease of analysis. The correctness of the protocol can be easily analyzed by *reachability analysis*. Protocols specified by CFSM models are also simple and easy to understand.

In CFSM model, each process is specified as a finite state machine. The protocol system is a set of machines: $M = [m_1, m_2, m_3, ...m_n]$. Between each pair of machines is a first-in-first-out (FIFO) queue in each direction, which represents the communication channel. A machine is specified as a set of states, a set of transitions, and a mapping between the states and transitions. The transitions include a *send-transition*, a *receive-transition*, and an *internal-transition*. A send-transition places the message at the end of an outgoing queue; the receive-transition takes the message from the front of an incoming queue; internal-transitions are those transitions upon which the machine does not change the contents of any queue. In order for a transition to take place, certain conditions must hold. For example, for a

receive-transition the message to be received should be present at the head of the incoming queue.

The protocol is defined with a diagram, often called *state-transition diagram* (or simply, *state diagram*). The states are given names or numbers, and are usually shown as circles. The possible transitions between states are indicated by pointed arcs with the transition stated alongside the arc [Ref. 3: p.118]. In the simplest form, the transitions are abbreviated and signed (a "-" indicating a send-transition, and a "+" indicating a receive-transition).

a. An Example Specification

As an example, the simple flow control method known as "stop and wait" protocol can be defined using a CFSM model.

The stop and wait protocol works as follows. There are two "machines:" the sender, and the receiver. Initially they are in a "ready" state. When the sender has a data frame to send, it transmits the frame and moves to a second state where it waits an acknowledgment from the receiver. The receiver next receives the frame and moves to a second state, from there sends an acknowledgment and returns to its initial state. The sender in turn, receives the acknowledgment and returns to its initial state.

The CFSM model of this scheme would be defined as a protocol machine PM = (S,M,I,T,C), where,

S = sets of states of machines = $\{S_1, S_2\}$

S_1 = states of m_1 = $\{0, 1\}$

S_2 = states of m_2 = $\{0, 1\}$

M = sets of messages = $\{M_{12}, M_{21}\}$

M_{12} = messages that can be sent from m_1 to m_2 = $\{D\}$

M_{21} = messages that can be sent from m_2 to m_1 = $\{A\}$

I = set of initial states of machines = $\{I_1, I_2\}$

I_1 = initial state of $m_1 = 0$

I_2 = initial state of $m_2 = 0$

T = partial transition function

$S_i \times \Sigma_i \mapsto S_j$, where,

$\Sigma_i = \{-x \mid x \in M_{ij}\} \cup \{+y \mid y \in M_{ji}\}$ i,j = 1,2

and,

$-x$ = sending of message x

$+y$ = receiving of message y

for m_1	for m_2
$0 \times -D \mapsto 1$	$0 \times +D \mapsto 1$
$1 \times +A \mapsto 0$	$1 \times -A \mapsto 0$

C = communication channels = $\{C_{12}, C_{21}\}$

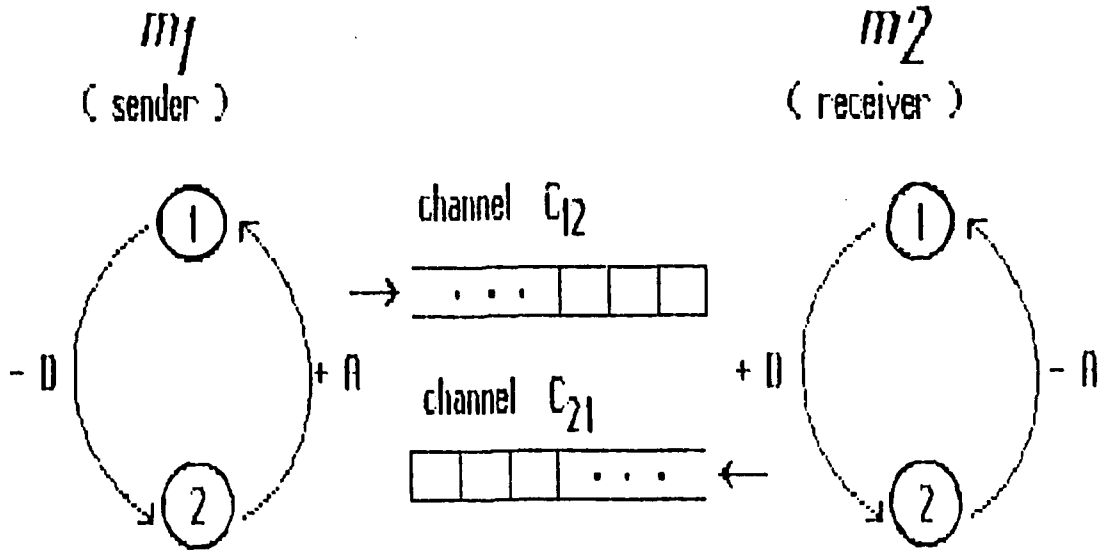
C_{12} = a FIFO queue connecting m_1 to m_2

C_{21} = a FIFO queue connecting m_2 to m_1

where, the contents of the queues are c_{ij} $(c_{12} \in M_{12}$, $c_{21} \in M_{21})$

[Ref. 8].

This definition is then illustrated as seen in Figure 1 on page 8, where m_1 and m_2 are shown as finite state machines with the communication channels in between. Note that this definition is taken in the simplest form to provide an example, and does not deal with all the aspects of the scheme (i.e., lost messages or acknowledgments are not considered here).



D = Data Frame
A = Acknowledgment

Figure 1. State Diagram for "Stop and Wait" Scheme

b. Reachability analysis

Reachability analysis is a common method used for analyzing the CFSM models. In this method the analysis is done by generating all possible *global states* from the initial global state. A global state is a tuple consisting of the states of each machine and the contents of each queue in the system. For the specification in Figure 1, this would be a 4-tuple,

$<(S_1,S_2)\ ,\ (Q_1,Q_2)>$

where,

S_1 = state of m_1

S_2 = state of m_2

Q_1 = contents of the queue C_{12}

Q_2 = contents of the queue C_{21}

The analysis starts with the initial global state, and a reachability graph is constructed by writing down the next possible global state(s) with an arc showing the transition which leads to that state. Figure 2 on page 10 shows the reachability graph for the above example.

The graph is generated as follows: in our example, initialiy both machines are in state0 and the queues are empty.

<(0 , 0) , (E , E)>

Inspecting the FSMs, there are two transitions that may take place. The sender may send a data frame, and the receiver may receive a data frame. Since the queues are empty (E), the receive-transition is not possible. The only possible transition is "-D." Thus the sender puts the data frame in C_{12} and moves to state1.

<(1 , 0) , (D , E)>

Again, two transitions are possible from this global state. The sender may receive an acknowledgment, and the receiver may receive a data frame. Since C_{21} is empty, the sender can not make the receive-transition. Thus, the receiver receives the data frame (essentially by reading and removing the data from the incoming queue). and moves to state1.

<(1 , 1) , (E , E)>

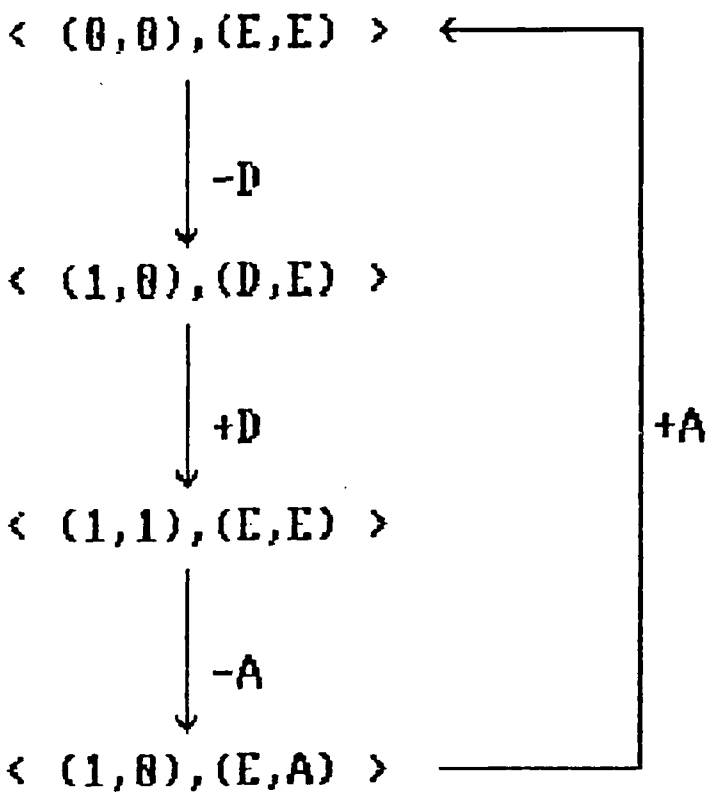


Figure 2. Reachability Graph for "Stop and Wait" Scheme

With similar reasoning, the only possible transition from this global state is "-A," which takes the receiver to state0.

$$<(1,0),(E,A)>$$

The next transition from here will be a "+A" upon which the sender moves to state0.

$$<(0,0),(E,E)>$$

This completes the analysis as the initial global state is reached again and the same sequence will follow. Thus the protocol has a cyclic behavior.

When doing the reachability analysis, we are actually searching for errors. This simple analysis shows that the defined scheme is free from certain types of errors.

The types of errors that can be detected by the analysis are:

- Deadlock state:

  This is a global state where all machines are in a receiving position (that is no send transitions leaving this state are specified), and all the queues are empty. Unless this is a predefined "final" global state, it is referred to as a deadlock state as there is an unexpected stop in the analysis.

- Unspecified reception:

  This is a state where at least one machine is in a receiving state but the message at the head of the incoming queue is not the message to be received.

- Non-executable transition:

  This is a transition specified in the state-diagram, which may never be executed by the protocol. This may not harm the function of the system but is a probable design error. Such transitions are placed in the system by the designer to take care of some combination of conditions that are predicted to occur. After the analysis those transitions are considered in the new design, and are either eliminated or corrections are done for making them executable.

2. Extended Finite State Machines

Although simple and easy to understand, the primary disadvantage of CFSM model is that "with no memory (other than the use of states) complex protocols with sequences can not be modeled and analyzed without a state explosion."[Ref. 9 : p.110] It is, in many cases, hard to determine whether the analysis will ever terminate. Moreover, for non-trivial protocols, it is impractical (if not impossible) to specify the protocol using a pure CFSM approach. Such shortfalls have led to a search for other methods. However, the numerous advantages that CFSM has over other methods attracted most of the studies towards methods which retain the strengths of CFSM technique while solving the weaknesses. The result of these studies can be grouped in

the Extended Finite State Machines (EFSM) category. Lundy and Miller [Ref. 10] propose an EFSM method which is referred to as *systems of communicating machines* (SCM).

The SCM model uses extended finite state machines with action tables and variables to specify the protocols. The variables are local (accessed by only a single machine) and shared (accessed by more than one machine). The machines communicate via shared variables. A communication channel can be a shared variable, or a machine which shares variables with the machines it is connecting. This obviously allows more control over the behavior of the channel. An example specification can be observed in Figure 3 on page 13, where a "sliding window protocol" with window size=2 is specified.

The analysis proposed for SCM is similar to reachability analysis, and is referred to as "system state analysis." This analysis method provides a significant reduction (compared to reachability analysis) in the number of states generated by the analysis. Details can be found in [Ref. 10].

Among other EFSM methods are Parallel Activity Specification Scheme (PASS), and Extended State Transition Language (ESTL) (also called "Estelle").

In PASS, the machines in the network are modeled as extended finite state machines with local variables. The PROLOG language is used for describing the semantics. [Ref. 11]

In ESTL, the protocol is modeled as a set of modules communicating with each other. The modules are specified as extended finite state machines by means of an extended PASCAL language. [Ref. 12]

The technique used in IEEE Standard 802.5 is another example of EFSM method. The protocol is specified by use of extended finite state machines, tables, and descriptive text.

3. Programming Languages

Methods using programming languages are more powerful than CFSM models in that they are very close to actual implementation. They also give the ability to model any protocol. However, they are more complex and difficult to

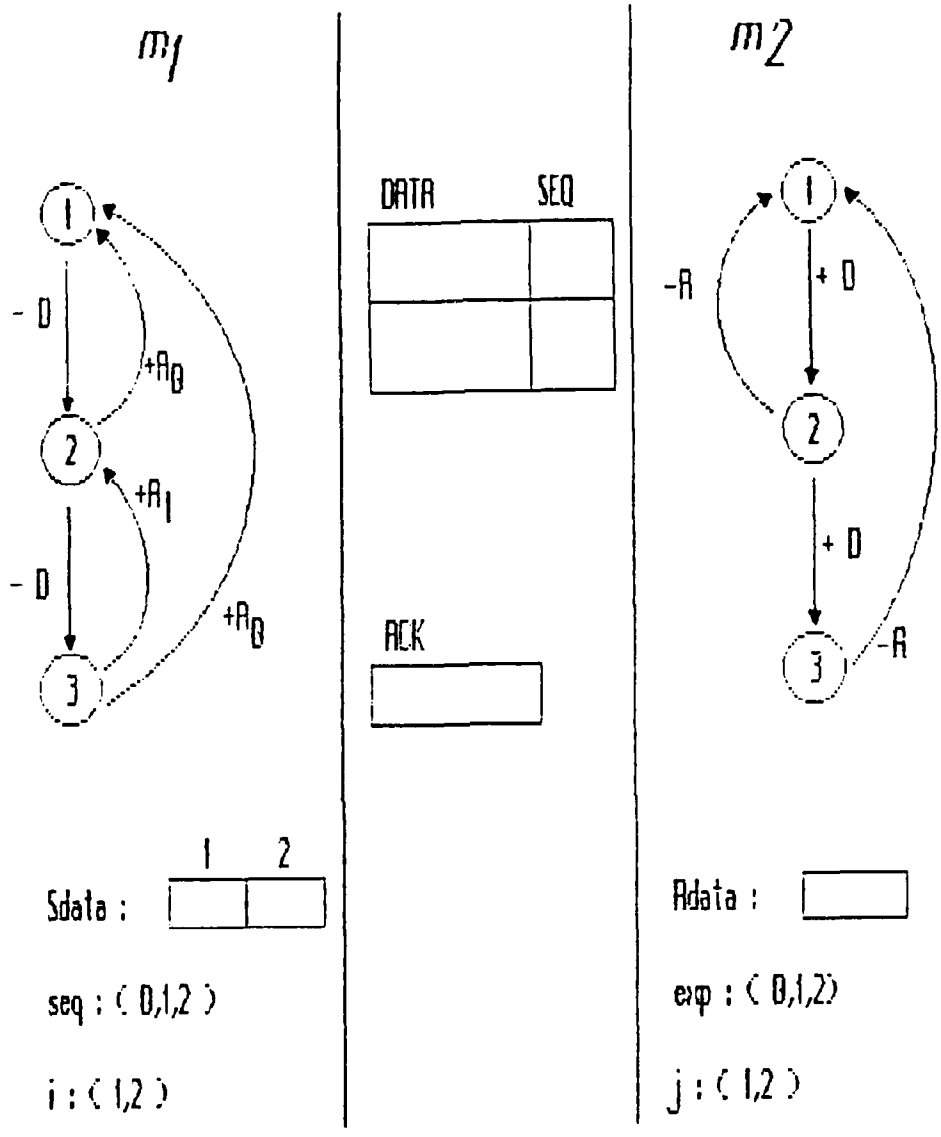


Figure 3. An Example of SCM Model

understand. As a result, the analysis of the protocol is more difficult. [Ref. 6 : p.10]

Communicating Sequential Processes (CSP) [Ref. 13], Language of Temporal Ordering Specification (LOTOS) [Ref. 14], Protocol Description and Implementation Language (PDIL) [Ref. 15] are some examples of languages

used for protocol specification and modeling. ADA has also been suggested for possible use as a protocol specification language [Ref. 16].

B. THE TOKEN RING NETWORK

As the name implies, the token ring network forms a closed path. Each station is connected to two others with unidirectional links (see Figure 4). A station receives from its upstream neighbor and transmits to its downstream neighbor. Data is transmitted as frames which contain addresses, control fields, and delimiters along with the actual information. Data transfer is sequential, bit by bit.

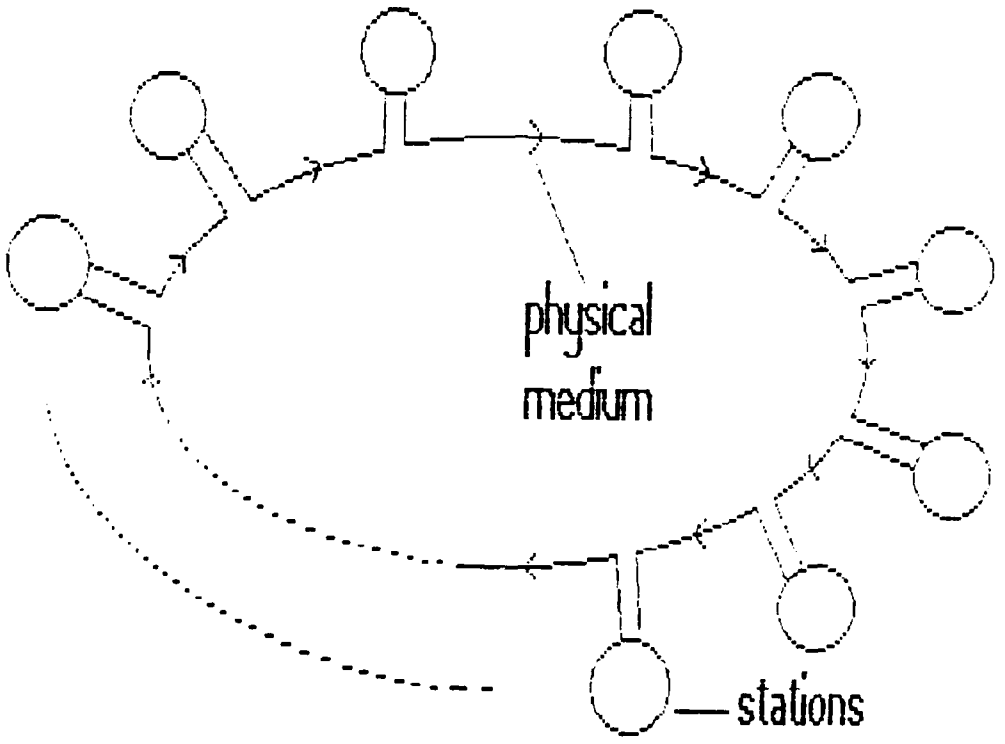


Figure 4. Ring Topology

Each station reads, regenerates (modifying if necessary), and retransmits each passing bit. The stations perform three major functions: putting the data on the ring, receiving data, and removing the data from the ring. The insertion of data is controlled by a token circulating inside the ring. In order to send a frame, a station first captures the token and modifies it to a frame format appending necessary fields and the information which is to be transmitted. [Ref. 5: P.344]

Since the ring is a closed loop, the frame will circulate around the ring "forever" unless it is removed. Thus, the removal of frame is a critical issue in ring protocols. Frames can be removed by the addressed station, or the originating station. However, the preferred way is that frames be removed by the originating station, thus allowing multiple stations to be addressed as well as automatic acknowledgment. [Ref. 5: P.344]

Once a station transmits a frame, the others read the passing frame bit by bit and retransmit to the next station. During this process, a station recognizing its own address in the destination address field will copy the frame while continuing to retransmit. Some of the bits may be altered for acknowledgement, error indication, or token reservation purposes.

When a station is transmitting, it no longer repeats the incoming bits; but checks them for certain fields. When the transmission of frames is completed, the originating station waits for the first frame it has transmitted. Upon recognizing the frame it has originated (by inspecting the source address), the station releases a new "free" token in the ring.

The above explanation is very general. The next chapter will go into more detail as specified in the standard.

III. A REVIEW OF THE IEEE STANDARD 802.5

The discussion, figures and tables provided in this chapter are cited from *ANSI/IEEE Standard 802.5-1985, Token Ring Access Method and Physical Layer Specifications* [Ref. 17]. The details which are irrelevant for the purposes of this study are omitted when appropriate.

The explanations reflect the author's interpretation of the standard. In some cases, other interpretations might be possible. The very fact that more than one interpretation might be possible, emphasizes the need for this study.

A. SCOPE

ANSI/IEEE Standard 802.5-1985, Token Ring Access Method and Physical Layer Specifications, is part of a family of standards for LANs. It defines a ring utilizing token-passing as the access method. The purpose of the standard is "compatible interconnection of data processing equipment via a local area network." The standard accomplishes the following:

- Definition of the frame formats,
- Definition of the medium access control (MAC) protocols,
- Description of the services provided by different sublayers to one another,
- Definition of the physical (PHY) layer functions,
- Definition of station attachments to the medium.

Of the above, this study is concerned with the MAC protocols section. Figure 5 on page 17 illustrates the sublayers, their relationship, and corresponding layers in the Open Systems Interconnection (OSI) Reference Model of the International Organization for Standardization (ISO).

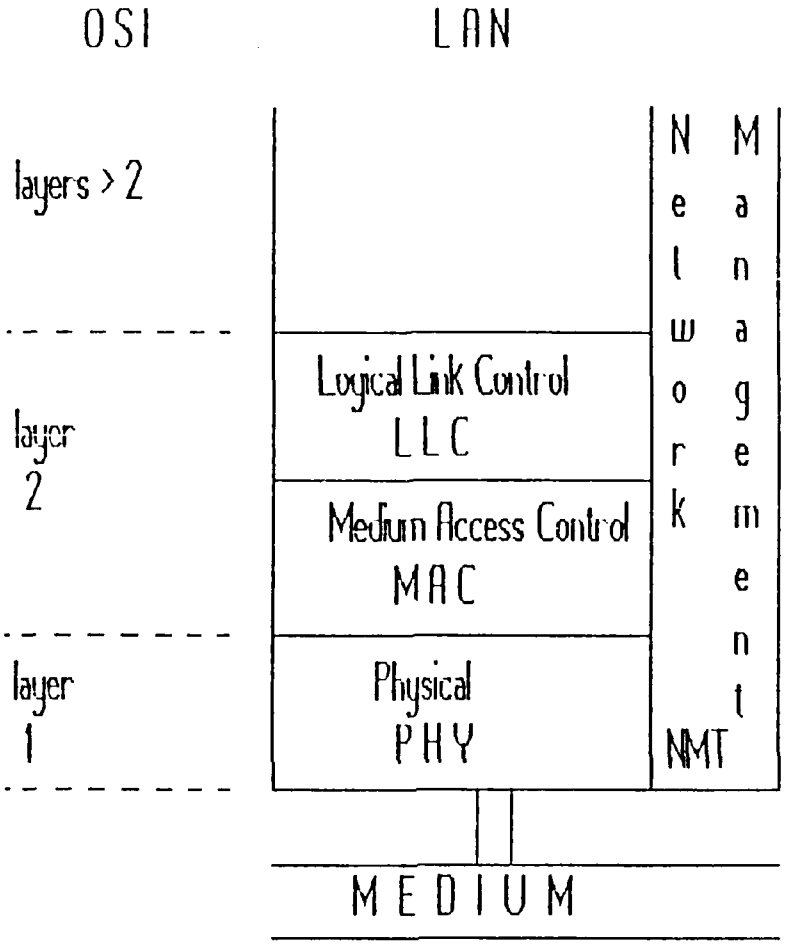


Figure 5. The LAN Model and Corresponding OSI Layers

B. FORMATS AND FACILITIES

The formats define basic structure of the transmissions on the ring. The formats and fields are transmitted starting with the left-most bit, the left-most bit being considered most significant.

Facilities include flags, registers, stacks, and timers. These are used for logic comparison, timing and error recovery purposes.

1. Formats

There are two types of frame formats which are used: token and frame. These formats are shown in Figure 6 on page 19. Figure 7 on page 20 provides a description of the fields. The token is the means by which "the right for transmitting frames" is controlled. A frame is the means by which data is transmitted.

The Starting Delimiter (SD) and the Ending Delimiter (ED) are used to mark the start and end of valid frames. The J and K bits are non-data bits which are simply an exception for the encoding scheme used in the medium[1] . The SD bit sequence is fixed. With ED, the first six bits are fixed while the other two bits serve different purposes. In a token these two bits are transmitted as "0"s. In a frame format the I bit is used to indicate the first (or intermediate) frame (1) or the last (or only) frame (0), and the E bit is used for error detection. The E bit is transmitted as "0" by the originating station, any other station which detects an error in the frame sets this bit to "1".

The Access Control (AC) field contains priority bits (PPP), a token bit (T), a monitor bit (M), and reservation bits (RRR). When in a token, priority bits show the priority of the frames that can be transmitted upon capturing the token. The token bit is used to discriminate between a frame and a token. A "0" indicates a token while a "1" indicates a frame. The monitor bit is used by the active monitor station to prevent a frame (or a token with priority greater than zero) from continuously circulating on the ring. The reservation bits allow stations to request the next token be issued at the priority level needed[2].

The Frame Control (FC) field defines the type of frame and some control functions. The F bits are frame type bits where a "00" indicates a MAC frame, and a "01" indicates a Logical Link Control (LLC) frame, the other two combinations "10" and "11" are reserved for future use. The Z bits are control bits used based on the type of frame.

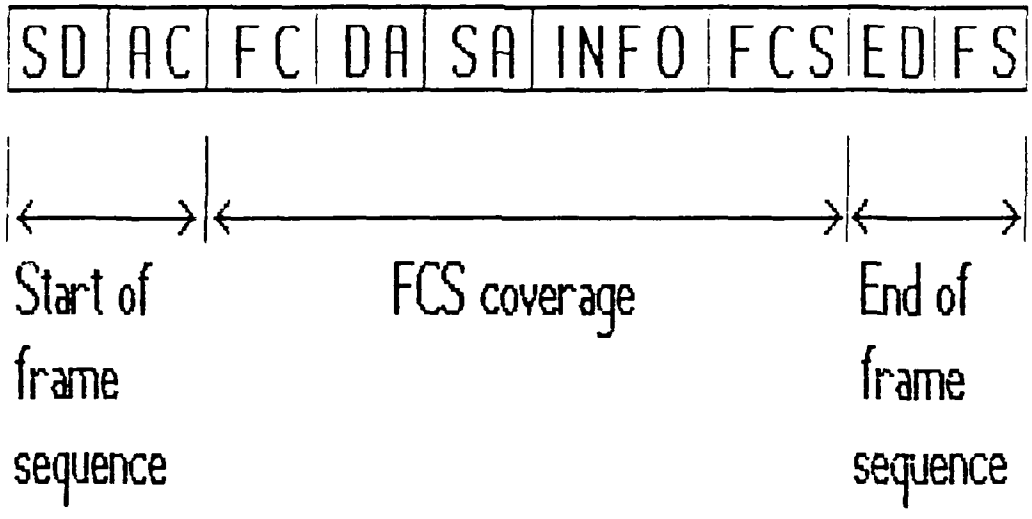
---

1 The encoding type is differential Manchester coding. Details can be found in "Physical Layer Specifications" section of the Standard.

2 A discussion of these bits is provided in section C.5 of this chapter.

a) Token

b) Frame



SD = Starting delimiter
AC = Access control
FC = Frame control
DA = Destination address
SA = Source address
INFO = Information
FCS = Frame check sequence
ED = Ending delimiter
FS = Frame status

Figure 6. Token Ring Formats

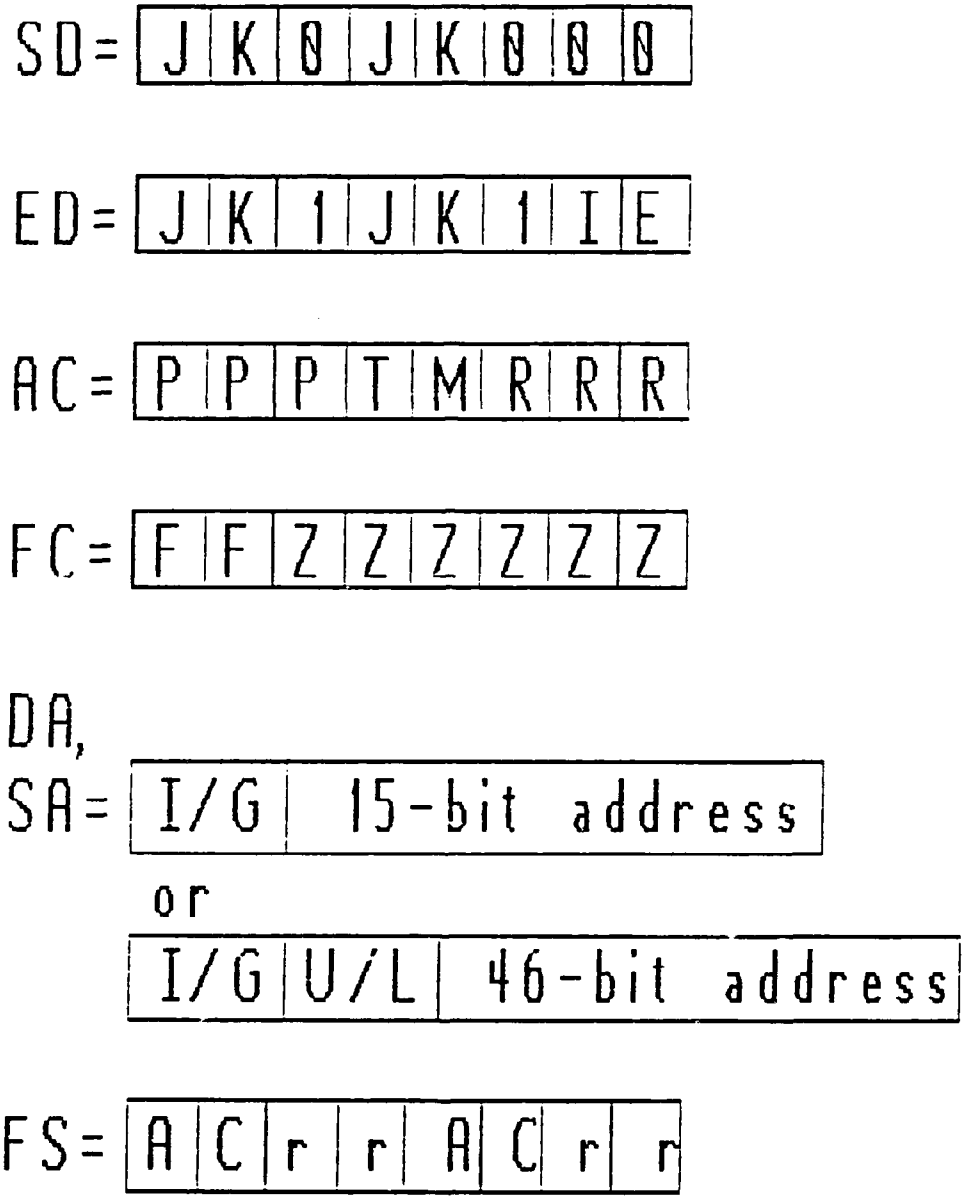


Figure 7. Field Descriptions

The Source Address (SA) and the Destination Address (DA) fields are used for indicating the source and destination of the frame. These fields can either be 16 or 48 bits in length, provided that they are of same length within a specific LAN. The I/G bit tells whether the address is an individual (0) or a group address (1). The I/G bit in an SA is always transmitted as a "0". A DA consisting of all "1"s indicates a broadcast address while a DA with all "0"s indicates a null address (frame not addressed for a particular station). For 48-bit addresses, the second bit (U/L) indicates whether the address is administered universally (0) or locally (1).

The Information (INFO) field contains zero, one, or more octets of user data. No maximum length is specified for the INFO field. However, the time required for the transmission of a frame can not exceed the token holding period specified for the station.

The Frame-Check Sequence (FCS) field is a 32-bit Cyclic Redundancy Check based on a standard generator polynomial of degree 32.

The Frame Status (FS) field is used for acknowledgement purposes. The A and C bits are transmitted as zeros by the originating station. The A bits are used as "address-recognized" indicator. A station which sees its own address in the DA field sets the A bits to "1"s. However, it may or may not copy the frame for some reason (i.e., the buffer is full); to indicate whether the frame is copied the C bits are used. If the frame is received (copied into buffer) by the destination station, it sets the C bits to "1"s. The "rr" bits are reserved for future use and are currently ignored by the repeaters.

2. Flags

Flags are used to "remember" the occurrence of particular events. There are three flags utilized in each station.

- *I_Flag:*

  Set upon receiving an ED with the I bit equal to zero.

- *SFS_Flag:*

  Set upon receiving a "Start of Frame Sequence".

- *MA_Flag:*

  Set upon receiving an SA which is equal to the station's own address.

3. Registers

Two registers are used to store the value of priority and reservation of the received AC field. These registers get updated each time an AC field is received.

- *Pr Register*

  Used for storing the priority of the most recently received AC field.

- *Rr Register*

  Used for storing the reservation of the most recently received AC field.

4. Stacks

There are two stacks which are used to keep track of priorities and to eventually bring the ring back to its original priority level when the priority has been raised. When transmitting a token, the station checks the Rr and Pm (priority of a queued PDU) to see if any of them is greater than the Pr, and if so transmits the token with a priority of the higher of Rr or Pm. At the same time, the station puts the value of Pr in *Sr Stack*, and the value of the priority of the token that was transmitted is put into *Sx Stack*. This will be clearer when the priority operation is explained in section C.5 below.

5. Timers

There are seven timers used in the standard. They are listed below with explanations of those that are relevant to this study.

- *Timer, Return to Repeat (TRR)*[3]:

  Used to ensure that the station returns to Repeat State after a given time period. The default value for TRR is 2.5 ms. However, this value has to be greater than the maximum ring latency, which is the signal propagation delay around a maximum 'ength ring plus the sum of all station delays.

- *Timer, Holding Token (THT)*:

---

3 The standard leaves the establishment of time-out values for these timers to the users of the ring.

Controls the maximum time period during which the station may transmit frames after capturing a token. The station can start the transmission of a frame only if it can be completed before the THT expires. Default value is 10ms.

- *Timer, Valid Transmission (TVX)*

  Used by the active monitor to detect the absence of valid transmissions. The time-out value is the sum of the time-out values of THT and TRR.

- *Timer, No Token (TNT)*

  Used by the stations to recover from token-related error situations. The time-out value is TRR plus *n* times THT (where *n* is the maximum number of stations on the ring).

- *Timer, Queue PDU (TQP)*
- *Timer, Active Monitor (TAM)*
- *Timer, Standby Monitor (TSM)*

C. TOKEN RING PROTOCOLS

Token ring protocols define the procedures used in the MAC layer.

1. Frame Transmission

Upon request of transmission of a *protocol data unit* (PDU), the Medium Access Control (MAC) unit puts the data in a frame format and enqueues it. The station then awaits for a proper token; a token with a priority less than or equal to the priority of the frame to be transmitted.

If a frame or a token with higher priority is circulating on the ring before the station can get a proper token, the station reads the reservation bits: if the value of the reservation bits is smaller than the priority of the awaiting frame, the reservation bits are modified to indicate the request for next token at desired priority level--otherwise the reservation bits are repeated unchanged. When the appropriate token is received, the station changes the token to a start of frame sequence while retransmitting, and stops repeating the rest of the token and starts transmission of the frame.

2. Token Transmission

When the transmission of the frame(s) is completed, the station inspects the MA_Flag to see whether its own address is returned in the SA

field. If MA_Flag is not set, the station transmits fill (a bit sequence of either "0"s, "1"s, or any combination of the two), until the flag is set. After the MA_Flag is set , the station generates a new token and puts this on the ring.

3. Stripping the Frames

Upon transmission of the new token, the station continues transmitting fill until the I_Flag is set; that is, the last frame (I=0) has returned. When the I-flag is set the station returns to repeat mode.

4. Frame Reception

When repeating the incoming bit stream, each station checks certain bits to see if they should only be repeated, or acted upon. If the frame-type bits (FF) in an FC field indicates a MAC frame, the control bits (Z bits) are interpreted by all stations on the ring.

If the DA field matches the station's own address, the station copies the rest of the frame (the FC, DA, SA, INFO, and FS fields) into a receive buffer: while continuing to repeat. The A and C bits in the FS field are modified as necessary before repeating to the next station.

5. Priority Operation

The P and R bits contained in the AC field work together to ensure that PDUs with higher priority than the current service priority of the ring are transmitted first, and all stations holding PDUs with the same priorities have equal rights for transmission.

When a station has priority PDU(s) ready for transmission, it modifies the R bits in the AC field as seen in Table 1.

Table 1. RESERVING THE NEXT TOKEN

condition	*modify RRR to;*
Pm > Rr	the value of Pm
Pm≤Rr	--unchanged--

After claiming the token, the station may transmit PDUs that are at or above the present ring service priority level. When it has completed the

transmissions (or when the THT expires), the station generates a new token. The priority of the new token is determined as shown in Table 2.

Table 2. MODIFYING THE TOKEN

condition	*transmit PPP as;*	*transmit RRR as;*
no PDUs with Pm > Pr, or no reservation with Rr > Pr	the value of Pr	the value of the greater of Pm or Rr
Pm > Pr or Rr > Pr	the value of the greater of Pm or Rr	000

When the second condition on Table 2 holds, the station that has raised the service priority of the ring with this procedure, becomes a "stacking station." From then on it has to monitor the token and lower its priority back to the old ring priority when suitable. The stacking station stores the old service priority as Sr and the new (transmitted) service priority as Sx.

The stacking station then claims every token with a priority equal to Sx, and takes the actions shown in Table 3.

Table 3. MODIFYING THE TOKEN (P = SX)

condition	*transmit PPP as;*	*transmit RRR as;*	*Sx Stack*	*Sr Stack*
Rr > Sr	the value of Rr	000	pop Sx, push (Rr)	--
Rr ≤ Sr	the value of Sr	the value of Rr	pop Sx	pop Sr

When the stacks are finally emptied as a result of stack-operations, the station discontinues its role as a stacking station.

6. Specification

The operation of the ring is described by finite state machine diagrams with additional tables and natural-language text supporting the diagrams. This

method is a type of EFSM method which was reviewed in the second chapter of this thesis.

There are three FSM diagrams used in the standard : Operational FSM Diagram, Standby Monitor FSM Diagram, and Active Monitor FSM diagram. These diagrams are reproduced in Appendix B.

Each station on the ring is a "dual FSM." The station may assume only one of the states in either the standby monitor FSM or the active monitor FSM at any given time. What makes the station a "dual FSM" is that, when in one of the following states in monitor FSMs, the station is also in one of the states in the operational FSM. The states which the station may assume at the same time with operational FSM are : state2 (INITIALIZE) or state4 (STANDBY) in standby monitor FSM, or state0 (ACTIVE) in active monitor FSM. When the station is in one of the other states in monitor FSMs (BYPASS, INSERTED, TX CL_TK, TX BEACON, TX FILL, TX PURGE), the activity of the operational FSM is suspended until transition is made to one of the previously mentioned states (INITIALIZE, STANDBY, ACTIVE) upon which the activity of the operational FSM is resumed at state0 (REPEAT).

Besides being a dual FSM, each station has two conceptual parts : a receive-side and a transmit-side. No matter which state the machine is in, the receive side takes the actions shown in Table 9 on page 48(Appendix B), according to the received bit stream. (See Appendix A for abbreviations and mnemonics.)

IV. PROBLEMS AND AMBIGUITIES

One of the primary purposes of a standard is to make sure the subject is presented clearly so that everyone using the standard interprets it identically.

When studying the standard, the author has come across some points which could be interpreted in more than one way, which were not clear, or which could be better explained. Those stated below are the ones which the author believes to be of importance.

A. OVERALL SPECIFICATION METHOD

Having studied some past approaches to formal protocol modeling and specification techniques, the author had expected that protocol modeling for this standard would follow a traditional approach. Although the method used in the standard appears to be a type of EFSM technique, this is not formally specified. Within the time that was devoted to this study, the author has not identified any source which includes the formal specification of this particular approach. Thus, any specification rule has to be searched in the standard itself, and that is not easy as any assumption or rule is established within the flow of the text whenever required rather than being specified as a whole in a separate section.

The FSM diagrams used in the standard are hard to grasp and different than the conventional illustration methods used in automata theory. Another observation is that the active monitor FSM and the standby monitor FSM have transitions to each other, which means that they actually are a complete single FSM. The simplicity gained by showing this single diagram as two separate diagrams is questionable.

The following sections discuss other problems, some of which result from the inadequacy of the method used; others are related to the logic behind the procedures.

B. PDU PRIORITIES

Pm is defined as the priority of a queued PDU, which is used as a basis for priority-operation. However, it is not clear in the standard whether this variable is kept in a stack or in a register. Pm is not declared as either. If there is a stack, then when do the values get "pushed" and "popped?" No stack operation has been defined on Pm. If Pm is kept in a register, then which PDU's priority does it represent? There may be more than one PDU in the queue, and the register can hold only one value at a time. Are there separate registers for each PDU in the queue which hold their Pm values?

C. FSM DIAGRAMS

In the generally accepted FSM notation, the states are shown as circles or ovals. The transitions are represented by pointed arcs between the states. An action normally involves a transition to another state (or to the same state), where the state represents a static situation and does not involve any actions.

The FSM diagrams illustrated in the standard have transitions "hidden" in most of the states, where they actually are packed into the names given to those states. One confusing example is discussed below in section C.1.

Even if the "machine" is not changing state, the transition(s) should be shown as a "loop" to the same state. The idea is to keep the supporting text as short as possible (without overloading the diagram) so that the user, once having read the text, can focus on the diagram without the need for referring the text over and over again. Besides, as required in the standard, in the case of discrepancy between FSM diagrams/tables and supporting text, the FSM diagrams/tables are given precedence. (Another reason to have better diagrams.)

1. Operational FSM : Transmission of PDUs

Transition01 to state1 (TX DATA_FR) is enabled if there is(are) PDU(s) queued and a token with $P \geq Pm$ is received. When this condition is satisfied, the station transmits an SFS and resets the THT and MA_Flag, thus making the transition to state1.

By definition, once in state1, the station transmits one or more frames as long as their Pm values are equal to or greater than the priority of the captured token (P) and THT has not expired.

However, the FSM does not exactly show this procedure. Looking at the FSM diagram, one can interpret the procedure as follows: the station makes the transition to state1 by transmitting an SFS and resetting THT and MA_Flag. Assuming nothing goes wrong, the only possible next transition is to state2. In order to make that transition, the predicate "PDU_END & (QUEUE_EMPTY V TEST_THT)" should hold. That is, the condition that "transmission of PDU[4] (which was initiated by transmission of the SFS) is completed, and the queue is empty or THT will expire before transmission of another PDU is completed," should be satisfied. In the case where there is a single PDU in the queue when the transition is made to state1, this transition will be enabled (PDU_END = TRUE, QUEUE_EMPTY = TRUE) and an EFS will be transmitted, TRR and I_Flag will be reset; thus the machine will move to state2.

a. Problem 1

Suppose there is more than one PDU in the queue and transmission of more than one PDU is possible before THT expires. Further, assume the token has priority P = 4 and PDUs have priorities (Pm = 4,3,2,.... etc.). First, the PDU with Pm = 4 will be transmitted. Then the predicate for transition to state2 will be tested (PDU_END = TRUE, QUEUE_EMPTY = FALSE, TEST_THT = FALSE): thus transition12 will not be enabled. The machine has to stay in state1 and transmit the next PDU in the queue (Pm = 3 ?). If we put the diagram aside and refer to the text, we will see that this is not allowed as only the PDUs with priority Pm≥P can be transmitted in this state. Therefore, the station will stay in state1 (doing nothing but testing the predicate) until the THT expires.

The need for testing Pm≥P condition (for each PDU) should be clearly delineated in the diagram.

---

4 The term PDU, as used here, refers to the portion of the frame between SFS and EFS.

b. Problem 2

Consider the above situation again, this time with the queued PDUs having priorities all at or above the captured token's priority (i.e., priorities are sufficient for the transmission of more than one PDU). After the transition to state1 is made and the transmission of the first PDU is completed, the referred predicate will be tested and again will not be enabled (PDU_END=TRUE, QUEUE_EMPTY=FALSE, TEST_THT=FALSE). Now, the next PDU satisfies the condition Pm≥P and will be transmitted. Because the transition to state2 is not made, no EFS for the previous frame has been transmitted. Furthermore, there is no SFS for the next frame, because the SFS is transmitted only once when making the transition to state1. The result is transmission of more than one PDU between a single SFS and a single EFS. Even if it was assumed that the name "TX DATA_FR" implies the transmission of complete frames, including an SFS and an EFS, there would still be a duplicate SFS and a duplicate EFS transmitted when making transition01 and transition12.

Besides showing the need for the test for Pm≥P condition for each frame, the diagram should further be improved to illustrate that an SFS-EFS pair is transmitted for each and every frame being transmitted.

2. Operational FSM : Modifying Stacks

When a stacking station receives a token with a priority value equal to the value of Sx and does not have a PDU with Pm > Sx, it makes the transition from state0 to state4 (transition03). The station does so by transmitting an SFS, popping Sx, and resetting TRR and SFS_Flag. In state4, the station transmits "0"s (to prevent the ring from being "idle") while modifying the stacks. Next, based on the value of Rr, a new token is transmitted and the transition is made to state5. When in state5, the station transmits fill and waits for the SFS_Flag to be set (or TRR to expire) upon which the transition is made to state0.

The procedure here seems rather extended. The question is whether transmission of SFS is essential, and whether one of the states could be eliminated.

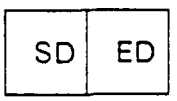
3. Active Monitor FSM : Stripping the Purge Frames

In the active monitor FSM, transition to state1 is from state2 where the station (active monitor) transmits purge frames in order to purge the ring prior to transmission of a new token. The station moves to state1 when the first purge frame has returned (FR_PURGE (SA=MA)). State1 exists "to ensure that all purge frames have been stripped from the ring before transmitting a new token."

It is questionable whether the predicate "TRR EXPIRED" is enough to assure that all the purge frames are stripped from the ring. Note that the problem "actions hidden in state names" also exists in states 1 and 2.

D. ABORT SEQUENCE

The abort sequence is defined as transmission of a starting delimiter and an ending delimiter :



This sequence is "used for the purpose of terminating the transmission of a frame prematurely." Should this abort sequence occur out of octet boundaries, the stations are required to be able to detect it.

There are three specific transitions shown in FSM diagrams, which require transmission of an abort sequence: transitions 11 and 43 in operational FSM, and transition02 in active monitor FSM. However, there are no predicates that utilize the receipt of an abort sequence. That is, the abort sequence is transmitted, but apparently causes no action.

There is not sufficient information about the receipt of an abort sequence--except that all the stations should be able to detect it anywhere within the incoming bit stream. What actions are to be taken upon receipt of an abort sequence? Further, in the case of transitions 11 and 43 of operational FSM, the station transmitting abort sequence makes the transition to state0 (REPEAT); then, who will strip this abort sequence off the ring?

V. SUGGESTED SOLUTIONS

A. OVERALL SPECIFICATION METHOD

The problem is that there is no formal definition of the protocol specification method used in the standard. If a previously suggested and formally specified method was not suitable for the purposes of this particular standard, then the method being used could have been specified in a separate section or in another publication. That would allow the user to first understand the methodology and then grasp the functional descriptions of the protocol with less confusion.

Given the method used, still better results could be achieved by trading natural-language text with tables and diagrams where possible. Often a table or diagram can describe a procedure more clearly and concisely than text. As an example, the "priority operation" explained using text in the standard [Ref. 17: pp.42-43] can be compared to the explanation relying on tables provided in Chapter III-Section C.5 of this thesis.

A suggestion regarding the FSM diagrams is that since the use of predicate-action notation on the diagram is constrained to space available, the predicates and actions could be placed in a table supporting the diagram. Appendix C includes FSM diagrams and action tables, to illustrate how they might appear. An illustration combining the active monitor FSM and the standby monitor FSM into a single FSM diagram is also presented in Appendix C.

Another suggestion concerning the "aesthetics" of the diagrams, is to avoid leaving actions in the names given to the states. It would make the diagrams more "self sufficient" if the states were left as static natures and any action that would not cause a state change was shown as a loop. Most of such portions of the diagrams are included in the solutions to other problems in the following sections. To provide an example, the "repeat" action in state0 of the operational FSM is included here. The fix proposed here is a minor change in

the "bit flipping loop state table." Since we already have that loop presentation, a few additions to predicate-action pairs, as illustrated below in Table 4, will be sufficient for our purposes.

Table 4. CHANGES TO "BIT FLIPPING LOOP STATE TABLE"

Transition	*Enabling Predicate*		*Action*
02	02A	PDU_QUEUED & (FR (R < Pm) V TK (P > Pm > R, P ≠ Sx))	SET R = Pm, REPEAT
	02B	FR_WITH_ERROR	SET E = 1, REPEAT
	02C	DA = MA (ADDRESS RECOGNIZED)	SET A = 1, REPEAT
	02D	FR_COPIED	SET C = 1, REPEAT
	02E	OTHERS	REPEAT

B. PDU PRIORITIES

The first question related with Pm is that it is not clear which PDU's priority it holds. Among the queued PDUs, the station would normally transmit the one with the highest priority first. Thus the Pm value used when reserving and using the token should hold a value which is equal to the priority of the PDU with the highest priority in the queue. This should not be left to users' intuition.

Another problem is whether the variable Pm is kept in a stack or a register. Using a stack seems impractical. It would require extra logic operations which otherwise would not be performed. The idea of a register on the other hand, raises the question as to when this register gets updated. Whenever a PDU is queued, its priority should be compared with the Pm value in the register; if the priority of the new PDU is greater than Pm, then this new (higher) value should be stored as Pm.

A still better way to avoid these questions could be stating that "the memory management scheme of the station shall be provided in such a way that the PDU with the highest priority shall be at the head of the PDU-queue,

the priority value of that PDU shall be used as Pm when making the necessary logic comparisons."

C. FSM DIAGRAMS

1. Solution 1 (Operational FSM)

Figure 8 illustrates how the test for Pm≥P condition for each PDU can be included in the FSM diagram.

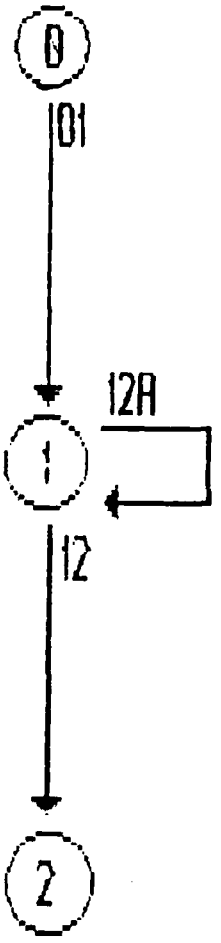


Figure 8. Partial FSM Diagram (Solution 1)

Here, the only change made to the original diagram is a loop which shows the transmission of PDUs, and numbered as "12A." The predicate for this transition is " PDU_END & (Pm≥P) & PDU_QUEUED & (¬TEST_THT)," and

the transition is taken by transmitting the PDU (TX_PDU). By this illustration it is more clearly seen that the condition Pm≥P is tested whenever a PDU is to be transmitted. However, the problem with multiple PDUs between a single SFS-EFS pair is still there. The next section proposes an improvement with use of additional states, which could provide a solution to this problem.

2. Solution 2 (Operational FSM)

The use of additional states (see Figure 9) can help clarify the illustration and avoid the interpretation that more than one PDU can be sent between a single SFS and a single EFS.

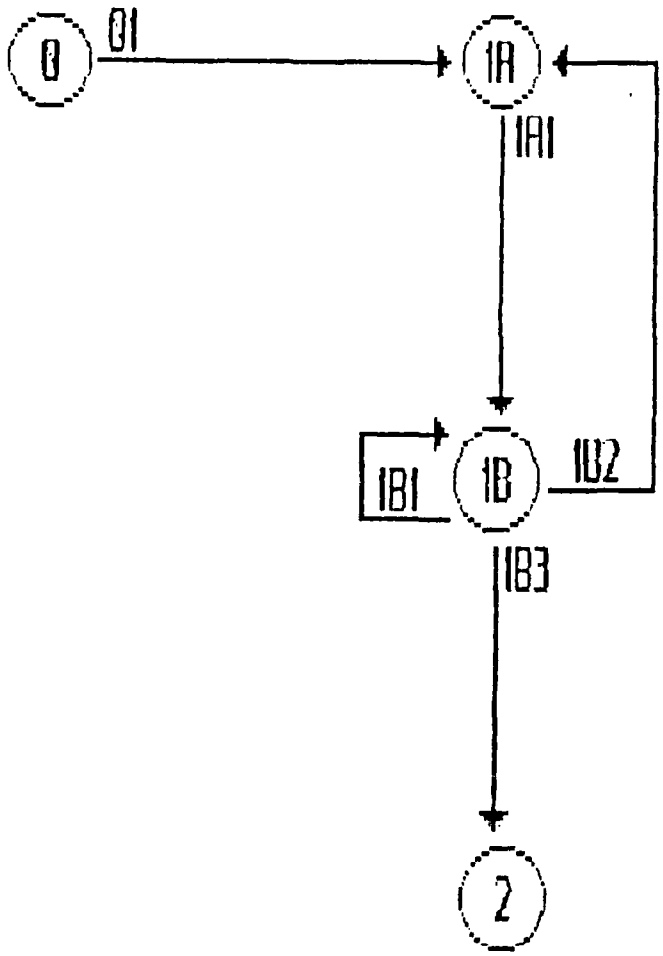


Figure 9. Partial FSM Diagram (Solution 2)

Table 5. PARTIAL ACTION TABLE (SOLUTION 2)

Transition	*Enabling Predicate*	*Action*
01	PDU_QUEUED & TK(P≤Pm)	RESET(THT, MA_FLAG)
1A1	TRUE	SFS(P=Pr, M=R=0)
1B1	¬PDU_END	TX PDU
1B2	(Pm ≥ P) & PDU_END & (¬TEST_THT)	EFS(I=1, E=A=C=0)
1B3	PDU_END & (QUEUE_EMPTY V (Pm<P) V TEST_THT)	EFS(I= E= A= C= 0), RESET (TRR, I_FLAG)
Note: When queue is empty, the comparison (Pm ≥ P) shall return the value "false."		

To trace this portion of the diagram, assume there are PDUs Pm=5,4,3,1 with a captured token P=4. Suppose THT will allow transmission of all these PDUs, provided that other conditions are satisfied. Transition01 will be enabled (PDU_QUEUED=TRUE, TK(P≤Pm)=TRUE), and the machine will move to state1A. The predicate for transition1A1 is always true, that is when in state1A transition to state1B will be made immediately by transmission of an SFS.

In state1B, the transmission of the PDU (Pm=5) will be made taking the transition-1B1. When the PDU is completely transmitted, PDU_END will become true and transition-1B1 will be disabled. Now, the predicates for transitions 1B2 and 1B3 will be tested. Transition1B3 can not be taken (QUEUE_EMPTY = FALSE). Predicate for transition1B2 is enabled (PDU_END=TRUE, Pm≥P=TRUE, ¬TEST_THT=TRUE). Thus an EFS (with I=1, E=A=C=0) will be transmitted and the machine will move to state1A.

From state1A, an SFS will be transmitted taking transition1A1, and the PDU (Pm=4) will be transmitted by taking the transition1B1. When transmission is completed, again the predicates for transitions 1B2 and 1B3 will be tested. This time, the predicate for 1B2 is disabled ((Pm=3)≥(P=4)

=FALSE), and the predicate 1B3 is enabled ((Pm=3) < (P=4) = TRUE). The EFS (with I=E=A=C=0) will be transmitted and the machine will move to state2.

3. Modifying the Stacks (Operational FSM)

The stack operations performed during the process include

- pop Sx (transition03)
- stack Sx (transition41) or pop Sr (transition42).

The rest of this discussion is based on the answer (or assumption) to the question "how long does it take a station to modify the stacks?" Modifying a stack simply requires the change of a pointer which points to some location in the memory, and in the case of a push operation an addition to that is a "write" operation. Assuming that the time required for that is trivial and the ring can tolerate the gap in transmission, the new token can be transmitted immediately without changing the bit stream to an SFS. This would eliminate state4 and state5, and there would be two transitions (from state0 looping back to itself) for stack modification purposes, which would be similar to transitions 41 and 42 (transition03 merged into those).

It appears, however, that the time needed to complete the stack operation is more than the ring can tolerate. There is still a question concerning the transmission of an SFS which later needs to be stripped, as to why it is not possible to transmit fill (or zeros) without an SFS, and transmit the new token when ready. Since the SD and part of the AC is already repeated before P=Sx can be detected, and thus an SFS can not be avoided; the question may be rephrased as "could the station just abort the old token, and transmit fill (or zeros) until the new token can be transmitted?"

The answer to that question seems to be related with TVX (Timer, Valid Transmission) and TNT (Timer, No Token). Leaving the ring without a token (or frame) for a certain amount of time might cause the TNT to expire, and standby monitors to take action. TNT (which actually is TRR + *n* THT, *n*=maximum number of stations) is not likely to expire before the station releases the new token--we are talking about not less than a 200ms of time in a maximum length

ring, which should be large enough when compared with execution speed of a processor at ns level. As for TVX, a similar reasoning is possible. TVX time-out value is the sum of THT and TRR timeout values. It can readily be observed from the operational FSM diagram that TRR timeout value alone gives enough time to the station to modify the stacks and release the new token--note that TRR is reset when taking transition03 and is considered to expire only after the transmission of the token, which is in transition51, and even in that case the strip may not be completed. Therefore it appears there should be no concern about aborting the old token and transmitting fill bits (regarding TNT or TVX).

A solution could be proposed to eliminate state5 as illustrated in Figure 10. This would also eliminate the need for an SFS_Flag ; as transitions 03 and 51 of the operational FSM are the only transitions where the SFS_Flag is utilized.

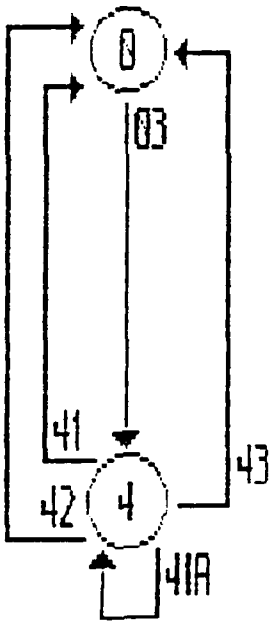


Figure 10. Partial FSM Diagram (Elimination of State5)

Table 6. PARTIAL ACTION TABLE (ELIMINATION OF STATE5)

Transition	*Enabling Predicate*	*Action*
03	(QUEUE_EMPTY V (PDU_QUEUED) & Pm < Sx)) & TK (P = Sx)	TX ABORT (I = 0), POP Sx, RESET (TRR, I_FLAG)
41	Rr > Sr	TK (P = Rr, M = R = 0), STACK Sx = P
41A	¬TOKEN_READY	TX ZEROS
42	Rr ≤ Sr	TK (P = Sr, M = 0, R = Rr), POP Sr
43	TOKEN_ERROR	STACK Sx = P

Note, however, that this solution is closely dependant on the function of the abort sequence. This point has led to the discussion of the abort sequence in the next section. As a result, it seems that the transmitted SFS and abort sequence have to be stripped off the ring for reasons discussed with abort sequence.

Given the above facts, it still is possible to eliminate the SFS_Flag and state5. Since state3 is serving a purpose similar to state5, by utilizing I_Flag: transitions from state4 can be made to state3, provided that the I bit in the ED of the abort sequence is transmitted as "0" and I_Flag is reset when taking transition03. This approach is illustrated in Figure 11 on page 40.

4. Stripping the Purge Frames (Active Monitor FSM)

The problem here is two-fold. First, does "TRR EXPIRED" take care of all the purge frames that were transmitted? Secondly, why should the station wait for the TRR to expire even if all the purge frames are received back before the TRR expires?

Apparently the time-out value of TRR is assumed to leave enough time to strip the purge frames. There still is a need to show the actions in states 1

and 2, and to exclude the need to wait for TRR to expire (even if all frames are stripped).

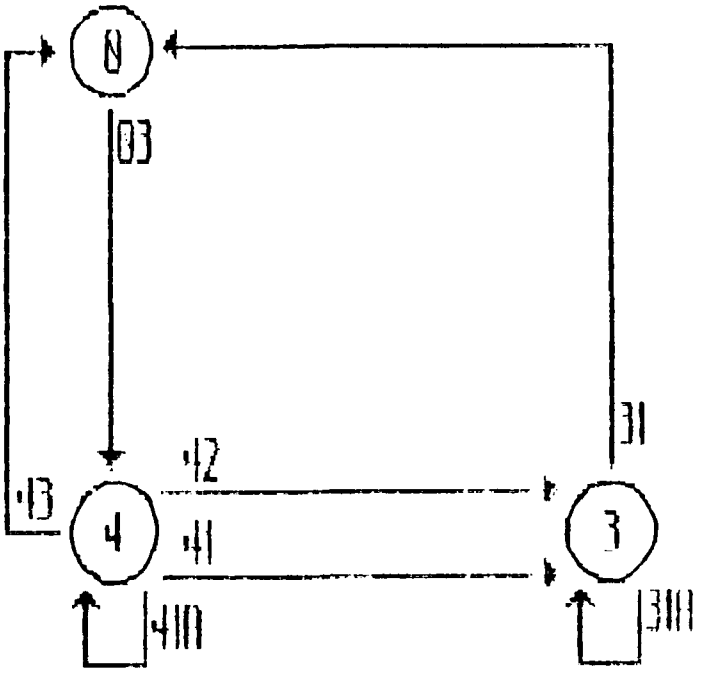


Figure 11. Partial FSM Diagram (Elimination of State5--final)

Table 7. PARTIAL ACTION TABLE (ELIMINATION OF STATE5--FINAL)

Transition	*Enabling Predicate*	*Action*
03	(QUEUE_EMPTY V (PDU_QUEUED) & Pm < Sx)) & TK (P = Sx)	TX ABORT (I = 0), POP Sx, RESET (TRR, I_FLAG)
31	I_FLAG SET V TRR_EXPIRED	
31A	(¬I_FLAG SET) & (¬TRR_EXPIRED)	TX FILL
41	Rr > Sr	TK (P = Rr, M = R = 0), STACK Sx = P
41A	¬TOKEN_READY	TX ZEROS
42	Rr ≤ Sr	TK (P = Sr, M = 0, R = Rr), POP Sr
43	TOKEN_ERROR	STACK Sx = P

The procedure here can be improved by taking an approach similar to the one used for PDU transmission in operational FSM. Figure 12 illustrates a solution for this issue.

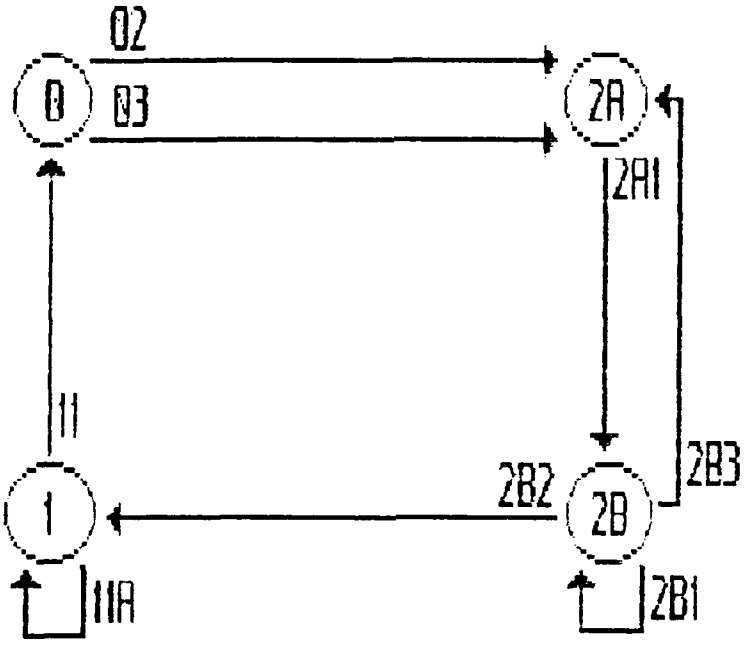


Figure 12. Improvement of Active Monitor FSM

Transitions 02 and 03 are the same as they are in the original diagram except the addition of "RESET MA_FLAG" to the actions. MA_Flag is used here to assure that at least one purge frame will be transmitted before the station moves to state1. State2 is divided into two states (2A and 2B). When in state2A, transition2A1 will be taken immediately by transmitting an SFS for the purge frame. In state2B, purge frame is transmitted by taking transition2B1.

After completion of the transmission, if MA_Flag is not set and TNT has not expired, an EFS (with I = 1) is transmitted; and the station moves to state2A to start another purge frame. When transmission of a frame is completed (in

state2B) and MA_Flag is set, the station transmits an EFS (with I=0), resets TNT and I_Flag, and moves to state1.

Table 8. PARTIAL ACTION TABLE (ACTIVE MONITOR)

Transition	*Enabling Predicate*	*Action*
11	I_FLAG SET V TRR_EXPIRED	TK (P=Rr, M=R=0), RESET (TVX, TAM), QUEUE AMP_PDU, STACK (Sx=P, Sr=0), MSI
11A	(¬I_FLAG SET) & (¬TRR_EXPIRED)	TX FILL
2A1	TRUE	SFS
2B1	(¬ FR_END) & (¬ MA_FLAG SET) & (¬ TNT_EXPIRED)	TX FRAME
2B2	FR_END & MA_FLAG SET	EFS (I=E=A=C=0)
2B3	FR_END & (¬ MA_FLAG SET) & (¬ TNT_EXPIRED)	EFS (I=1, E=A=C=0)

In state1, fill bits are transmitted via transition11A until the I_Flag is set (all transmitted frames are stripped) or TRR has expired; upon which the station releases a new token taking transition11.

D. ABORT SEQUENCE

There are two questions related to abort sequence. The first one is what actions are to be taken upon receipt of an abort sequence, and the second one is which station removes the abort sequence off the ring.

Observing the operational FSM it is seen that the abort sequence is transmitted by a station which has previously captured the token and is in a transmit-state (as opposed to repeat). Since transmitting the abort sequence takes that station to repeat state, and since no token will be present on the ring, all the stations will be in their repeat-state. Thus the answer to the first question is that the action to be taken upon receipt of an abort sequence is to repeat it to the next station. (This problem does not exist when a frame is

aborted by transition02 of active monitor FSM, as the station (active monitor) aborting the frame moves to a transmit-state.)

The second question is related to stripping the abort sequence. When the FSM diagrams are inspected, it is seen that one of the two possible things may happen. The first possibility is that a pure SD-ED pair will cause the TVX to expire since TVX is reset only when a token or frame with M bit of the AC field equal to zero is received. When the TVX expires, the active monitor will take transition03 and eventually transmit a new token. Another possibility is that the abort sequence might come following an SFS (as opposed to pure SD-ED) and thus have an AC field. The first time it is repeated by the active monitor the M bit will be set, and the second reception of this bit stream by the active monitor will enable transition02 of active monitor FSM; thus the active monitor will again release a new token when appropriate.

These aspects of the abort sequence could have been included in its definition (i.e., "Transmission of an abort sequence causes the active monitor to purge the ring and release a new token.")

VI. CONCLUSIONS AND RECOMMENDATIONS

This thesis studies the formal protocol specification and analysis techniques with an emphasis on a recently written protocol standard. A brief discussion of protocol specification techniques is provided. The token ring access method is reviewed both in general form and the way it is specified in the IEEE Standard 802.5.

Some problems found with the standard are stated and possible solutions to those are suggested. The FSM illustrations are found to be inadequate in that they might lead to misinterpretations. Three such problems with the operational FSM related to PDU transmissions and one with the active monitor FSM concerning transmission and strip of the purge frames are discussed and improvements are suggested. Other minor problems with the definitions of PDU priorities and the abort sequence are also presented.

Towards the improvement of FSM illustrations, use of action tables supporting the diagrams is suggested and examples are provided.

The major concern is the specification method used in the standard. Most of the problems (or questions that potential users might ask) are a result of the method being used. This method is a combination of extended automata and natural-language text. A formal definition is not available for this method, and is not likely to be established given the complexity arising from the use of the natural-language. When the wide spectrum of the users is considered, it is not possible to have a clear specification that would address all the users with this method.

The problems pointed out in this thesis could be of vital importance to the proper use of the standard. They also indicate probable reasons to consider other specification techniques in the future standards, if not in the current ones.

Potential research subjects in this area include the following:

- Development of a formally specified model.
- Specification of the Token Ring Protocol using a previously suggested formal specification method.
- Analysis of the protocol with this new specification.
- Study of the current protocol with a validation of the findings of this thesis.
- Similar studies with other protocols and standards.
- Application issues related to implementation or use of particular network products to specific communications needs; studying the options and selection criteria along with maintenance/expansion/management issues of the selected technology.

APPENDIX A. ABBREVIATIONS AND MNEMONICS USED IN THE STANDARD

A = Address-Recognized Bit
AMP = Active Monitor Present
BCN = Beacon
C = Frame Copied Bit
CL = Claim
DA = Destination Address
DAT = Duplicate Address Test
E = Error Detected Bit
ED = Ending Delimiter
EFS = End-of-Frame Sequence
FR = Frame
FS = Frame Status (field)
I = Intermediate Frame Bit
M = Monitor Bit
MA = My (station's) Address
MSI = MA_STATUS.indication
NMT = Network Management
P = Priority (of the AC)
PDU = Protocol Data Unit
Pm = PDU Priority
Pr = Last Priority Value Received
PRG = Purge
R = Reservation (of the AC)
Rr = Last Reservation Value Received
RUA = Received Upstream Neighbor's Address
SA = Source Address
SFS = Start-of-Frame Sequence

SMP = Standby Monitor Present

Sr = Highest Stacked Received Priority

SUA = Stored Upstream Neighbor's Address

Sx = Highest Stacked Transmitted Priority

TAM = Timer, Active Monitor

THT = Timer, Holding Token

TK = Token

TNT = Timer, No Token

TQP = Timer, Queue PDU

TRR = Timer, Return to Repeat

TSM = Timer, Standby Monitor

TVX = Timer, Valid Transmission

TX = Transmit

TK(P=x, M=y, R=z) = Token with P=x, M=y, and R=z

FR(P=x, M=y, R=z) = Frame with P=x, M=y, and R=z

& = AND

¬ = Boolean Not

V = OR

/ = the greater of

APPENDIX B. FSM DIAGRAMS AND TABLES USED IN THE STANDARD

Table 9. RECEIVE ACTION TABLE

Received	*Action*
REPORT FRAME STATUS	MSI
TK(P < Sx)	CLEAR STACKS
SA = MA	SET MA_FLAG
TOKEN V FRAME	STORE (Pr, Rr)
I = 0	SET I_FLAG
SFS	SET SFS_FLAG
FR (SA = MA, RUA ≠ SUA)	MSI

Properties of a frame:

1. Is bounded by a valid SD and ED
2. Has the E bit equal to 0
3. Is an integral number of octets in length
4. Is composed of only 0 and 1 bits between the SD and ED
5. Has the FF bits of the FC field equal to 00 or 01
6. Has a valid FCS
7. Has a minimum of 10 (16 bit addressing) or 18 (48 bit addressing) octets between SD and ED

REPORT FRAME STATUS:

- 1 & 2 & 3 & 4 & 5 & 6 & 7
- 1 & ¬2 & 3 & 4 & 5 & 6 & 7
- 1 & 2 & (¬3 V ¬4 V (5 & ¬6) V (5 & ¬7))

Figure 13. Properties of a Frame, and "Report Frame" Conditions

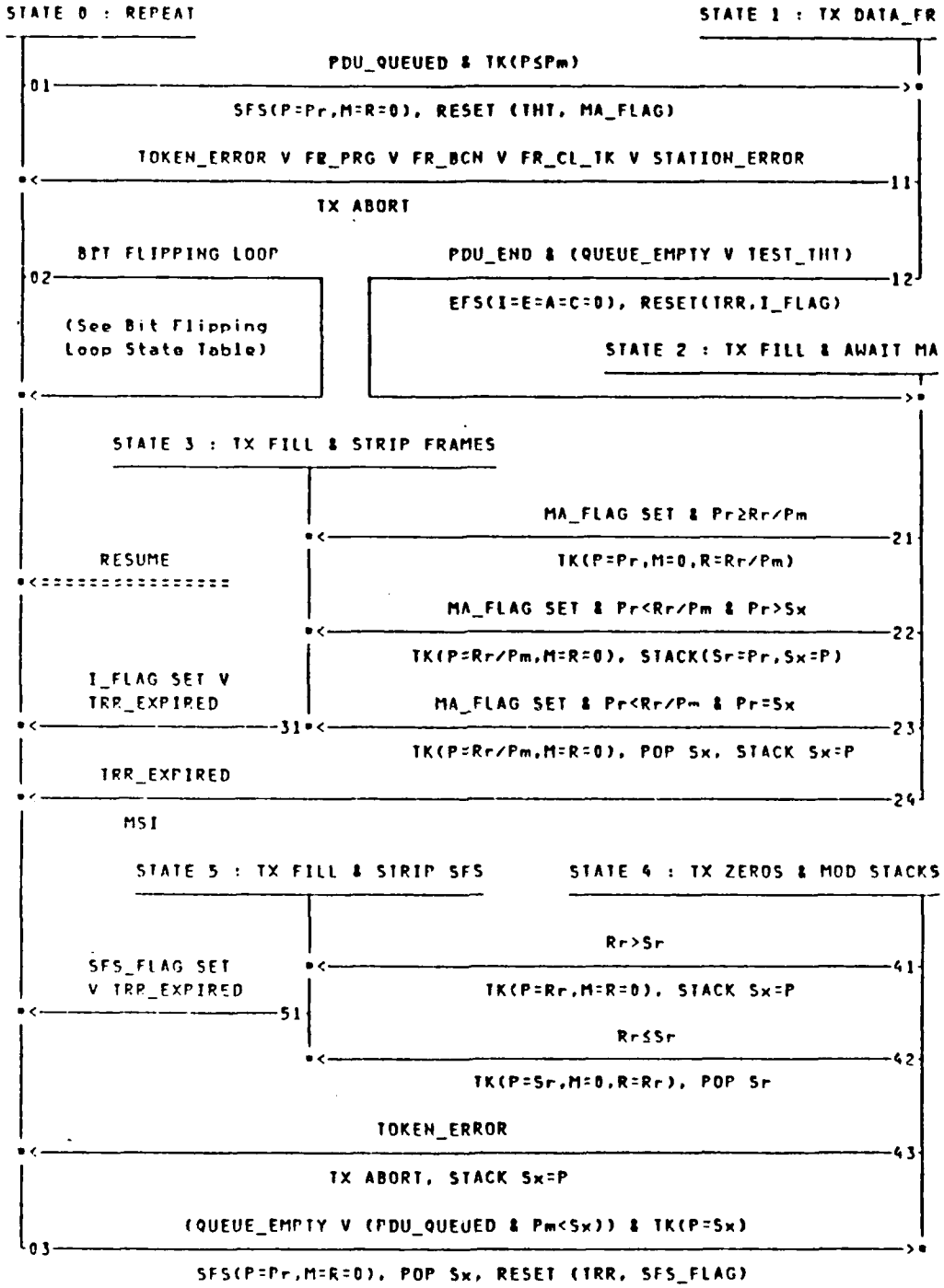


Figure 14. Operational FSM Diagram

REF	INPUT	OUTPUT
02A	PDU_QUEUED &(FR(R<Pm) V TK(P>Pm>R,P≠Sx)	SET R=Pm
02B	FR_WITH_ERROR	SET E=1
02C	DA=MA (ADDRESS RECOGNIZED)	SET A=1
02D	FR_COPIED	SET C=1

Figure 15. Bit Flipping Loop State Table

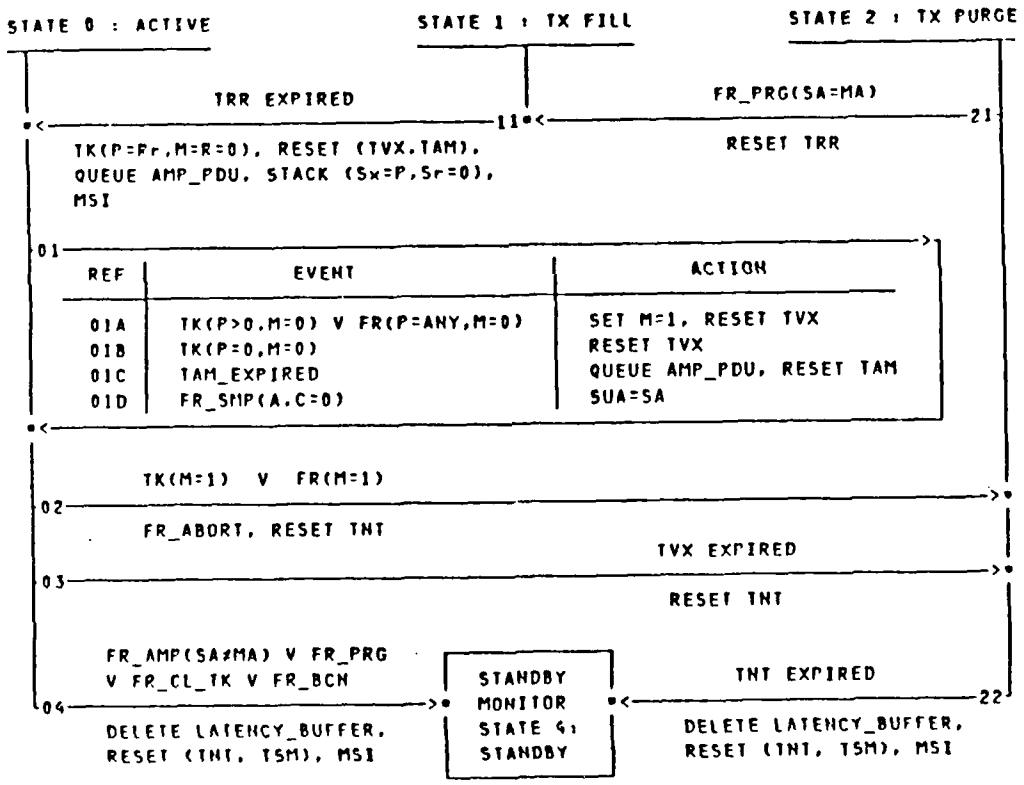


Figure 16. Active Monitor FSM Diagram

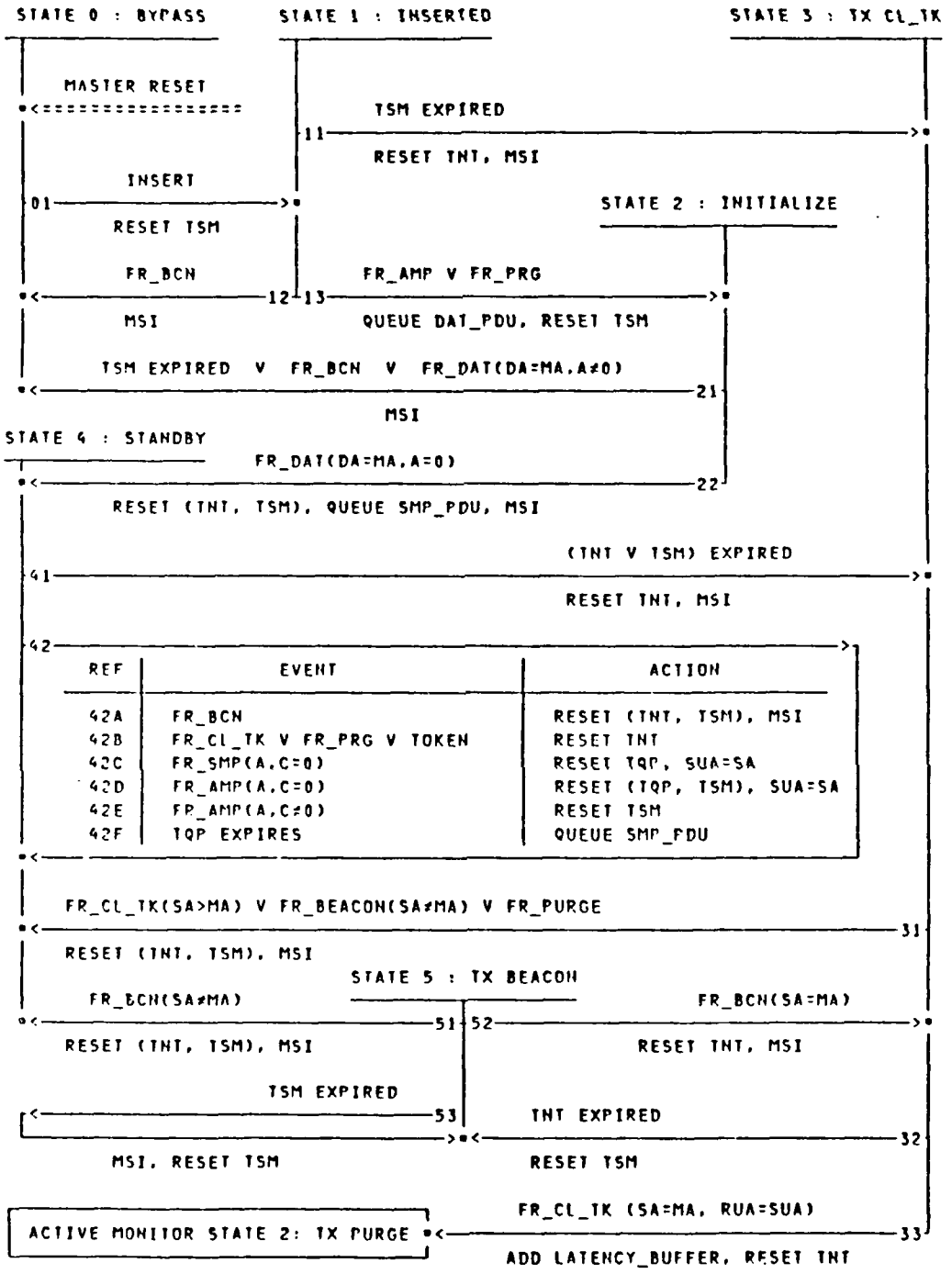
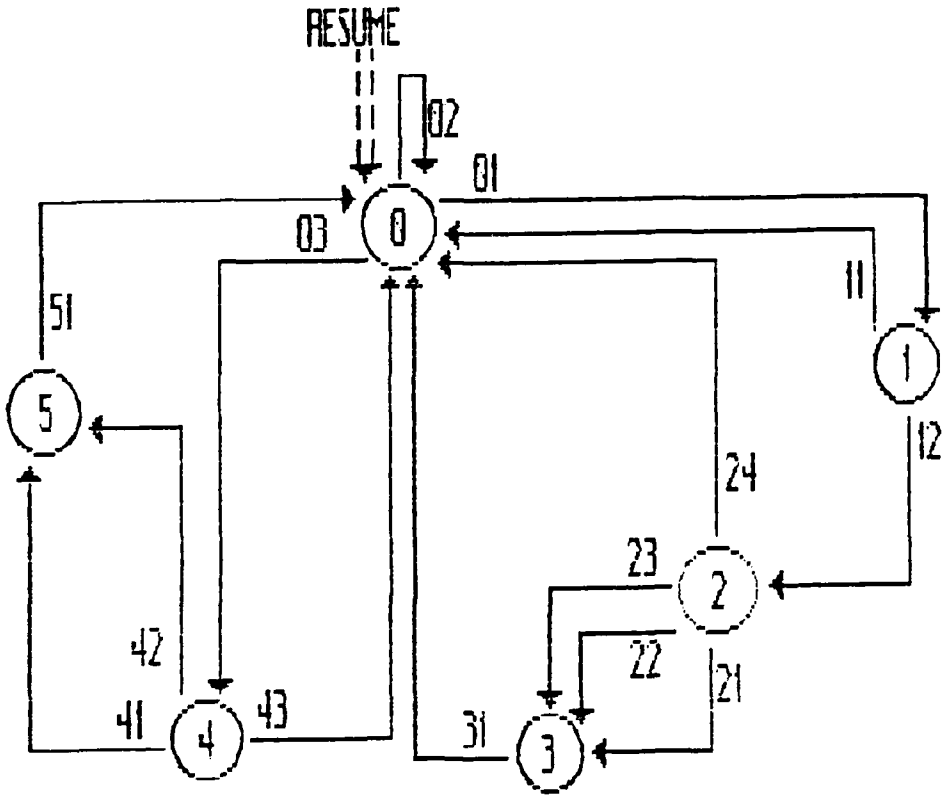


Figure 17. Standby Monitor FSM Diagram

APPENDIX C. SUGGESTED USE OF FSM WITH ACTION TABLE



STATE 0 = REPEAT
STATE 1 = TX DATA_FR
STATE 2 = TX FILL & AWAIT MA
STATE 3 = TX FILL & STRIP FRAMES
STATE 4 = TX ZEROS & MOD STACKS
STATE 5 = TX FILL & STRIP SFS

Figure 18. Operational FSM Diagram

Table 10. ACTION TABLE (OPERATIONAL FSM)

Transition	*Enabling Predicate*		*Action*
01	PDU_QUEUED & TK(P≤Pm)		SFS(P=Pr, M=R=0), RESET(THT, MA_FLAG)
02	02A	PDU_QUEUED & (FR(R̄<Pm) V TK(P>Pm>R, P ≠ Sx))	SET R=Pm
	02B	FR_WITH_ERROR	SET E=1
	02C	DA=MA (ADDRESS RECOGNIZED)	SET A=1
	02D	FR_COPIED	SET C=1
03	(QUEUE_EMPTY V (PDU_QUEUED & Pm<Sx)) & TK(P=Sx)		SFS(P=Pr, M=R=0), RESET(TRR, SFS_FLAG)
11	TOKEN_ERROR V FR_PRG V FR_BCN V FR_CL_TK V STATION_ERROR		TX ABORT
12	PDU_END & (QUEUE_EMPTY V TEST_THT)		EFS(I= E= A= C= 0), RESET (TRR, I_FLAG)
21	MA_FLAG SET & Pr ≥ Rr/Pm		TK(P=Pr, M=0, R=Rr/Pm)
22	MA_FLAG SET & Pr<Rr/Pm & Pr>Sx		TK(P= Rr/Pm, M=R=0), STACK(Sr=Pr, Sx=P)
23	MA_FLAG SET & Pr<Rr/Pm & Pr=Sx		TK(P= Rr/Pm, M=R=0), POP Sx, STACK Sx=P
24	TRR EXPIRED		MSI
31	I_FLAG SET V TRR_EXPIRED		
41	Rr>Sr		TK(P=Rr, M=R=0), STACK Sx=P
42	Rr ≤ Sr		TK(P=Sr, M=0, R=Rr), POP Sr
43	TOKEN_ERROR		TX ABORT, STACK Sx=P
51	SFS_FLAG SET V TRR_EXPIRED		

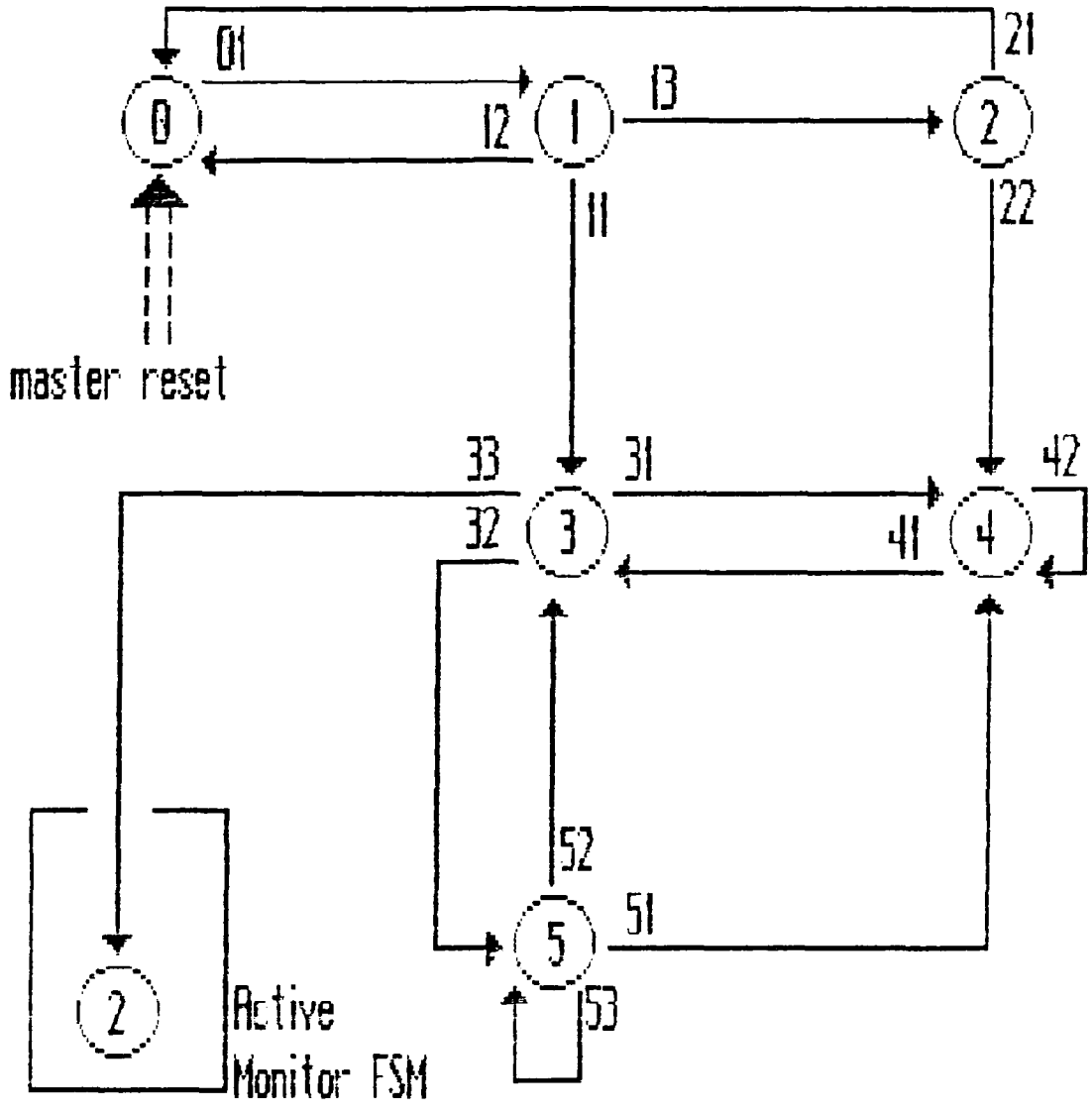
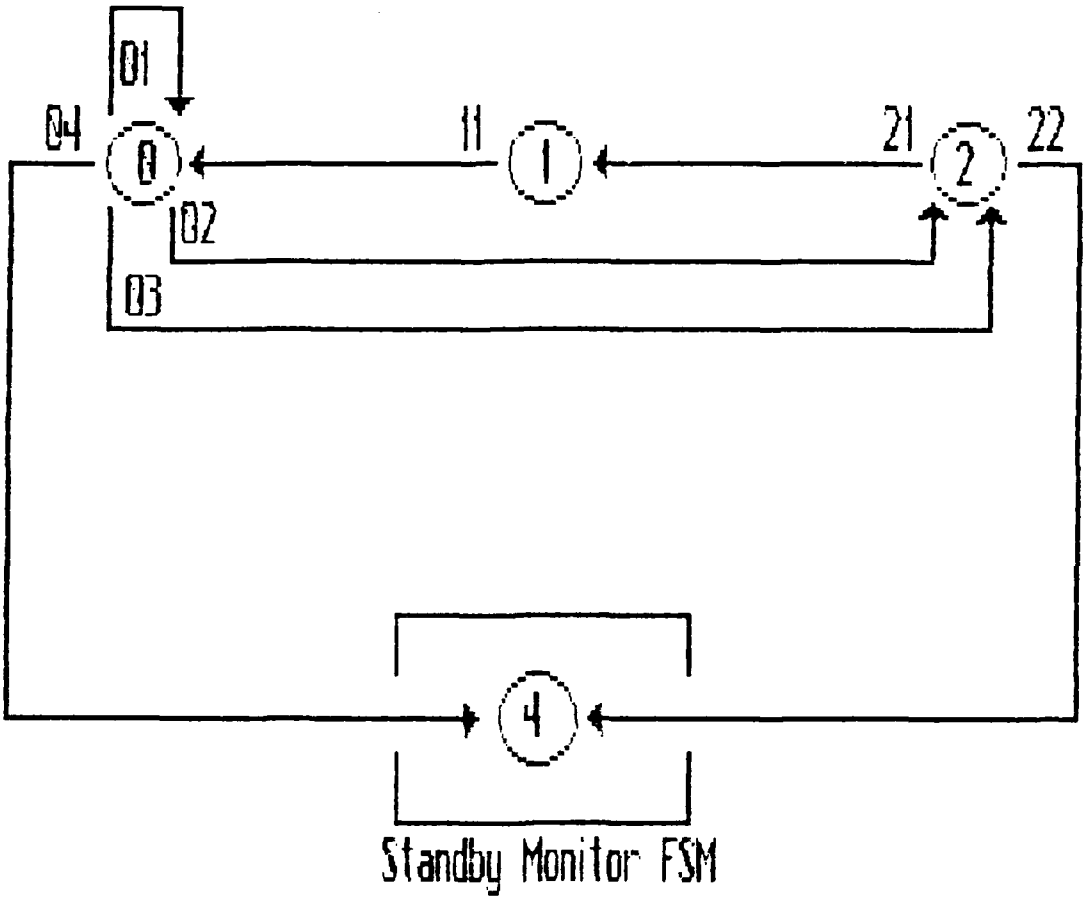


Figure 19. Standby Monitor FSM Diagram

Table 11. ACTION TABLE (STANDBY MONITOR FSM)

Transition	*Enabling Predicate*		*Action*
01	INSERT		RESET TSM
11	TSM EXPIRED		RESET TNT, MSI
12	FR_BCN		MSI
13	FR_AMP V FR_PRG		QUEUE DAT_PDU, RESET TSM
21	TSM EXPIRED V FR_BCN V FR_DAT (DA = MA, A ≠ 0)		MSI
22	FR_DAT (DA = MA, A = 0)		RESET (TNT, TSM), QUEUE SMP_PDU, MSI
31	FR_CL_TK (SA > MA) V FR_BCN (SA ≠ MA) V FR_PRG		RESET (TNT, TSM), MSI
32	TNT_EXPIRED		RESET TSM
33	FR_CL_TK (SA = MA, RUA = SUA)		ADD LATENCY_BUFFER, RESET TNT
41	(TNT V TSM) EXPIRED		RESET TNT, MSI
42	42A	FR_BCN	RESET (TNT, TSM), MSI
	42B	FR_CL_TK V FR_PRG V TOKEN	RESET TNT
	42C	FR_SMP (A, C = 0)	RESET TQP, SUA = SA
	42D	FR_AMP (A, C = 0)	RESET (TQP, TSM), SUA = SA
	42E	FR_AMP (A, C ≠ 0)	RESET TSM
	42F	TQP EXPIRED	QUEUE SMP_PDU
51	FR_BCN (SA ≠ MA)		RESET (TNT, TSM), MSI
52	FR_BCN (SA = MA)		RESET TNT, MSI
53	TSM EXPIRED		MSI, RESET TSM



STATE 0 = ACTIVE
STATE 1 = TX FILL
STATE 2 = TX PURGE

Figure 20. Active Monitor FSM Diagram

Table 12. ACTION TABLE (ACTIVE MONITOR FSM)

Transition	*Enabling Predicate*		*Action*
01	01A	TK(P > 0, M = 0) V FR(P = ANY, M = 0)	SET M = 1, RESET TVX
	01B	TK(P = 0, M = 0)	RESET TVX
	01C	TAM EXPIRED	QUEUE AMP_PDU, RESET TAM
	01D	FR_SMP (A, C = 0)	SUA = SA
02	TK(M = 1) V FR(M = 1)		FR_ABORT, RESET TNT
03	TVX EXPIRED		RESET TNT
04	FR_AMP (SA ≠ MA) V FR_PRG V FR_CL_TK V FR_BCN		DELETE LATENCY_BUFFER, RESET (TNT, TSM), MSI
11	TRR EXPIRED		TK(P = Rr, M = R = 0), RESET (TVX, TAM), QUEUE AMP_PDU, STACK (Sx = P, Sr = 0), MSI
21	FR_PRG (SA = MA)		RESET TRR
22	TNT EXPIRED		DELETE LATENCY_BUFFER, RESET (TNT, TSM), MSI

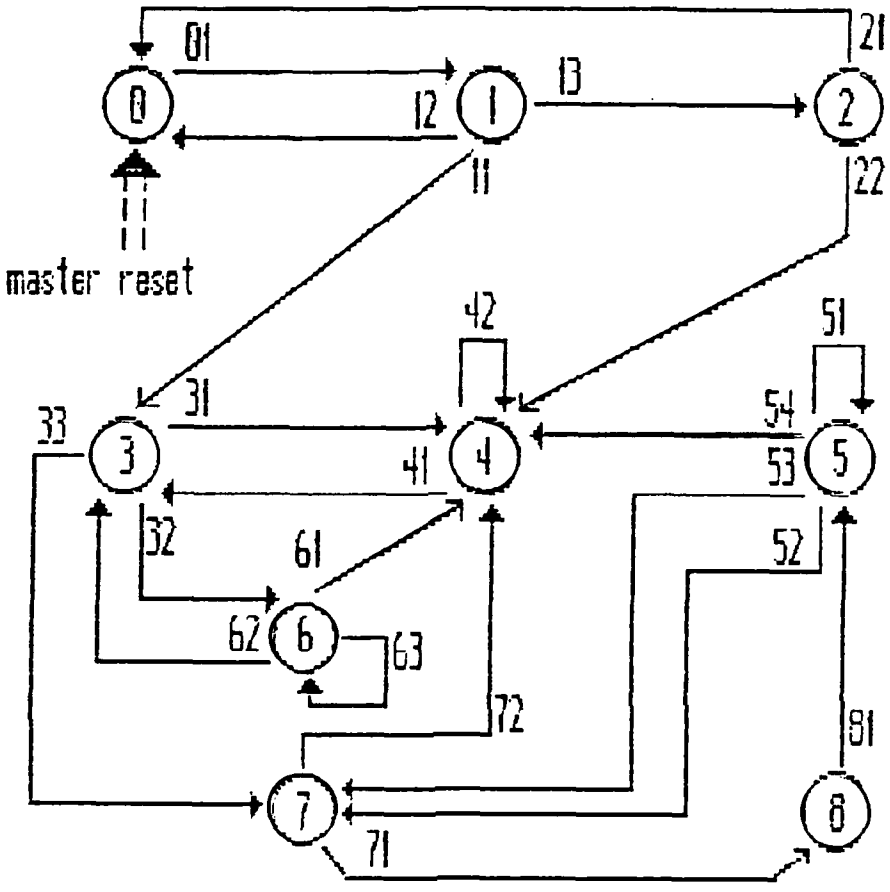


Figure 21. Combined Monitor FSM Diagram

Table 13. ACTION TABLE (COMBINED MONITOR FSM)

Transition	*Enabling Predicate*		*Action*
01	INSERT		RESET TSM
11	TSM EXPIRED		RESET TNT, MSI
12	FR_BCN		MSI
13	FR_AMP V FR_PRG		QUEUE DAT_PDU, RESET TSM
21	TSM EXPIRED V FR_BCN V FR_DAT (DA = MA, A ≠ 0)		MSI
22	FR_DAT (DA = MA, A = 0)		RESET (TNT, TSM), QUEUE SMP_PDU, MSI
31	FR_CL_TK (SA > MA) V FR_BCN (SA ≠ MA) V FR_PRG		RESET (TNT, TSM), MSI
32	TNT_EXPIRED		RESET TSM
33	FR_CL_TK (SA = MA, RUA = SUA)		ADD LATENCY_BUFFER, RESET TNT
41	(TNT V TSM) EXPIRED		RESET TNT, MSI
42	42A	FR_BCN	RESET (TNT, TSM), MSI
	42B	FR_CL_TK V FR_PRG V TOKEN	RESET TNT
	42C	FR_SMP (A, C = 0)	RESET TQP, SUA = SA
	42D	FR_AMP (A, C = 0)	RESET (TQP, TSM), SUA = SA
	42E	FR_AMP (A, C ≠ 0)	RESET TSM
	42F	TQP EXPIRED	QUEUE SMP_PDU
51	51A	TK(P > 0, M = 0) V FR(P = ANY, M = 0)	SET M = 1, RESET TVX
	51B	TK(P = 0, M = 0)	RESET TVX
	51C	TAM EXPIRED	QUEUE AMP_PDU, RESET TAM
	51D	FR_SMP (A, C = 0)	SUA = SA
52	TK(M = 1) V FR(M = 1)		FR_ABORT, RESET TNT
53	TVX EXPIRED		RESET TNT
54	FR_AMP (SA ≠ MA) V FR_PRG V FR_CL_TK V FR_BCN		DELETE LATENCY_BUFFER, RESET (TNT, TSM), MSI

61	FR_BCN (SA ≠ MA)	RESET (TNT, TSM), MSI
62	FR_BCN (SA = MA)	RESET TNT, MSI
63	TSM EXPIRED	MSI, RESET TSM
71	FR_PRG (SA = MA)	RESET TRR
72	TNT EXPIRED	DELETE LATENCY_BUFFER, RESET (TNT, TSM), MSI
81	TRR EXPIRED	TK(P = Rr, M = R = 0), RESET (TVX, TAM), QUEUE AMP_PDU, STACK (Sx = P, Sr = 0), MSI

LIST OF REFERENCES

1. Schneidewind, N.F., Notes, presented in the Computer Networks (Wide Area/Loca Area) course, Naval Postgraduate School, Monterey, CA, July-September 1988.

2. Lundy, G.M., Notes, presented in the Data and Computer Communications course, Naval Postgraduate School, Monterey, CA, October-December 1988.

3. Halsall, F., *Data Communications, Computer Networks and OSI*, England : Addison-Wesley, 1988.

4. Madron, W.T., *Local Area Networks: The Second Generation*,, New York: John Wiley & Sons, 1988.

5. Stallings, W., *Data and Computer Communications*, New York : Macmillan, 1988.

6. Lundy, G.M., *Systems of Communicating Machines : A Model for Communication Protocols*, Ph.D. Thesis, School of Information and Computer Science, Georgia Institute of Technology, Atlanta, GA, 1988.

7. Rudin, H., "An Informal Overview of Formal Protocol Specification," *IEEE Communications Magazine*, v.23 No.3, pp.46-52, March 1985.

8. Miller, R. E., "The Communicating Finite Machine Model for Communication Protocols," paper, presented in the Formal Modeling of Computer Network Protocols course, Naval Postgraduate School, Monterey, CA, January-March 1989.

9. Miller, R.E., and Lundy, G.M., "An Approach to Modeling Communication Protocols Using Finite State Machines and Shared Variables," IEEE Global Telecommunications Conference, Houston, TX, December 1-4, 1986.

10. Lundy, G.M., and Miller, R.E., *Specification and Analysis of a Data Transfer Protocol Using Systems of Communicating Machines*, School of Information and Computer Science, Georgia Institute of Technology, Atlanta, GA, October 1, 1988.

11. Fleischmann, A., *PASS : A Technique for Specifying Communication Protocols*, Protocol Specification, Testing and Verification VII, North-Holland, 1987.

12. Linn, R. J., "The Features and Facilities of Estelle : A Formal Description Technique Based Upon an Extended Finite State Machine Model," *Procedings of the Fifth Internaticnal Workshop on Protocol Specification, Testing and Verification*, Toulouse-Moissac, France, June 10-13,1985.

13. Hoare, C.A.R., "Communicating Sequential Processes," *Communications of the ACM*, v.21, August 1978.

14. Leduc, G.J., "The Intertwining of Data Types and Processes in LOTOS," *Protocol Specification, Testing and Verification VII*, North-Holland, 1987.

15. Ansart, J.P., and others, "Description, Simulation and Implementation of Communication Protocols Using PDIL," *Proceedings of the 1983 ACM SIGCOMM Conference on Communications Architectures and Protocols*, March 8-9, 1983.

16. Castanet, R., Dupuex, A., and Guitton, P. "Ada, a Well-suited Language for the Specification and Implementation of Protocols," *Proceedings of the*

Fifth International Workshop on Protocol Specification, Testing and Verification, Toulouse-Moissac, France, June 10-13, 1985.

17. Institute of Electrical and Electronics Engineers, *ANSI/IEEE Std.802.5-1985, Token Ring Access Method and Physical Layer Specifications*, 1985.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Commander, Naval Telecommunications Command Naval Telecommunications Command Headquarters 4401 Massachusetts Avenue, N.W. Washington, D.C. 20390-5290	1
4.	Kara Kuvvetleri Komutanligi Kutuphane, Bakanliklar, Ankara, TURKEY	1
5.	Kara Harp Okulu Komutanligi Kutuphane, Bakanliklar, Ankara, TURKEY	1
6.	Department of Computer Science Naval Postgraduate School, Attn: Prof. G.M. Lundy, Code 52Ln Monterey, CA 93943-5000	4
7.	Department of Administrative Sciences Naval Postgraduate School, Attn: Prof. W. Gates, Code 54Gt Monterey, CA 93943-5000	1
8.	Nejdet AYIK P.K. 25. 57001, Sinop, TURKEY	2
9.	Department of Administrative Sciences Naval Postgraduate School, Attn: Prof. D.C. Boger, Code 54Bo Monterey, CA 93943-5000	1

10. Ortadogu Teknik Universitesi 1
Kutuphane,
Ankara, TURKEY

11. Bogazici Teknik Universitesi 1
Kutuphane,
Istanbul, TURKEY